USER MANUAL

Accessory 65M

UR Protected/OPTO (Sourcing 24 in 24 out)

3Ax-603740-xUxx

November 19, 2013

DELTA TAU
Data Systems, Inc.
NEW IDEAS IN MOTION ...

21314 Lassen St. Chatsworth, CA 91311 // Tel. (818) 998-2095 Fax. (818) 998-7807 // www.deltatau.com

Copyright Information

© 2013 Delta Tau Data Systems, Inc. All rights reserved.
This document is furnished for the customers of Delta Tau Data Systems, Inc. Other uses are unauthorized without written permission of Delta Tau Data Systems, Inc.
Information contained in this manual may be updated from time-to-time due to product improvements, etc., and may not conform in every respect to former issues.
To report errors or inconsistencies, call or email:

Delta Tau Data Systems, Inc. Technical Support
Phone: (818) 717-5656
Fax: (818) 998-7807
Email: support@deltatau.com
Website: http://www.deltatau.com

Operating Conditions

All Delta Tau Data Systems, Inc. motion controller products, accessories, and amplifiers contain static sensitive components that can be damaged by incorrect handling. When installing or handling Delta Tau Data Systems, Inc. products, avoid contact with highly insulated materials. Only qualified personnel should be allowed to handle this equipment.

In the case of industrial applications, we expect our products to be protected from hazardous or conductive materials and/or environments that could cause harm to the controller by damaging components or causing electrical shorts. When our products are used in an industrial environment, install them into an industrial electrical cabinet or industrial PC to protect them from excessive or corrosive moisture, abnormal ambient temperatures, and conductive materials. If Delta Tau Data Systems, Inc. products are directly exposed to hazardous or conductive materials and/or environments, we cannot guarantee their operation.

WARNING

A Warning identifies hazards that could result in personal injury or death. It precedes the discussion of interest.

Caution

A Caution identifies hazards that could result in equipment damage. It precedes the discussion of interest.

Note

A Note identifies information critical to the understanding or use of the equipment. It follows the discussion of interest.

REVISION HISTORY				
REV.	**DESCRIPTION**	**DATE**	**CHG**	**APPVD**
1	Update manual for new release: new 24 V connector	2/20/07	C.P	R.N
2	Updated 16-bit ADC option	12/9/09	C.P	S.F
3	Completely revised manual	12/17/12	DCDP	R.N
4	Reformatted entire manual Added Power PMAC3/PMAC2	11/15/2013	R.N	R.N

Table of Contents

INTRODUCTION **6**

SPECIFICATIONS **7**

Part Number 7

Environmental Specifications 8

Electrical Specifications 8

Physical layout, Mounting 9

USING THE ACC-65M WITH POWER PMAC3 ..... **10**

Step 1: Preparing the Ring Controller 11

Step 2: MACRO ASCII Communication 13

Step 3: Finishing up the ACC-65M Setup 15

Step 4: I/O Data Registers 16

Step 5: Using the ACC-65M Data 17

Digital Inputs and Outputs *19*

Analog Inputs (ADCs) and Outputs (DACs) *20*

Using the ADCs for Servo Feedback *22*

General Purpose Relay Outputs *23*

USING THE ACC-65M WITH POWER PMAC2 ..... **25**

Step 1: Preparing the Ring Controller 26

Step 2: MACRO ASCII Communication 27

Step 3: Finishing up the ACC-65M Setup 29

Step 4: I/O Data Registers 30

Step 5: Using the ACC-65M Data 31

Digital Inputs and Outputs *33*

Analog Inputs (ADCs) and Outputs (DACs) *35*

Using the ADCs for Servo Feedback *38*

General Purpose Relays *39*

USING THE ACC-65M WITH TURBO PMAC2 ..... **41**

Step 1: Preparing the Ring Controller 42

Step 2: MACRO ASCII Communication 44

Step 3: Finishing up the ACC-65M Setup 46

Step 4: I/O Data Registers 47

Nodes and Addressing *47*

Turbo Ring Controller I/O Node Registers *48*

Step 5: Using the ACC-65M Data 49

Digital Inputs and Outputs *49*

Analog Inputs (ADCS) *52*

Using the ADCs for Servo Feedback *54*
Analog Outputs (DACs) *55*
General Purpose Relay Outputs *57*

CONNECTOR PINOUTS AND WIRING 59
24 VDC Input 59
Digital Inputs 60
Wiring the digital Inputs *61*
Digital Outputs 62
Wiring the digital outputs *63*
Analog Connector 64
Wiring the Analog (ADC) Inputs *65*
Wiring the Analog (DAC) Outputs *66*
Wiring the General Purpose Relays *66*
MACRO Connection 68
Universal Serial Bus (USB) 69

TROUBLESHOOTING 70
Initializing the ACC-65M, Clearing Faults 70
Error Codes (7-Segment LED) 71
LED Status 72
Input and Output LED Indicators *72*
Status LED *72*
Relay Status LED *72*
MACRO Link LED *72*

APPENDIX A: MEMORY MAP 73
PMAC3 Style ASIC 73
Using the ACC-65M with PMAC3 Address Offsets *74*
PMAC2 Style ASIC 75
Using the ACC-65M with PMAC2 Address Offsets *76*

APPENDIX B: E-POINT JUMPERS 77

APPENDIX C: SCHEMATICS 78

INTRODUCTION

The accessory 65M (ACC-65M) is a boxed MACRO peripheral I/O module with 24 isolated, self-protected, digital inputs and 24 isolated, self-protected, digital outputs. The ACC-65M is typically configured as a slave in a MACRO ring via either fiber optic or RJ-45 connection.

The inputs can be either sinking or sourcing depending on the user's wiring.

The outputs are strictly sourcing up to 600 mA per channel.

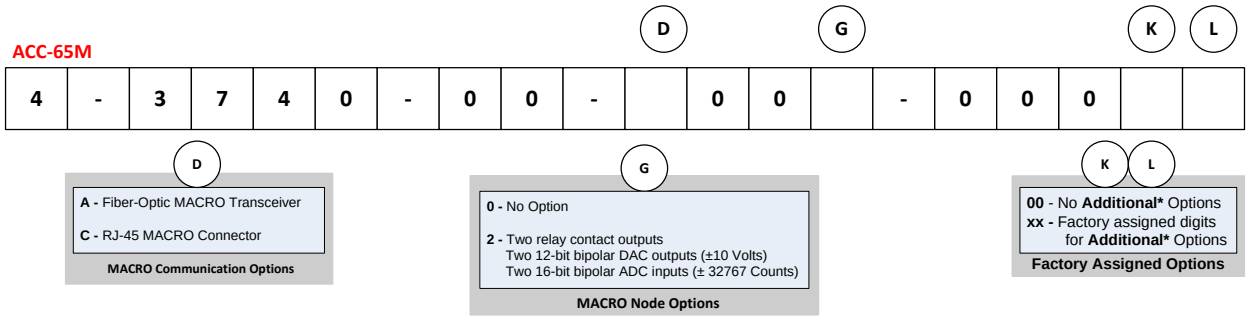
Optional sets of two analog inputs, two analog outputs and two general purpose relay contacts are available.

The ACC-65M is compatible with the following Delta Tau controllers:

- All Turbo PMAC2 board-level MACRO cards
- Turbo PMAC2 Ethernet Ultralite
- Power or Turbo UMAC with ACC-5E
- Power or Turbo Brick family (equipped with the MACRO option)
- Power UMAC with ACC-5E3
- Power PMAC EtherLite

SPECIFICATIONS

Part Number



The possible part number configurations are:

Options	Part Number
Fiber optic connectors	4-3740-00-A000-00000
RJ-45 connectors	4-3740-00-C000-00000
Fiber optic connectors 2 x 16-bit bipolar ADC analog inputs (±10 VDC) 2 x 12-bit bipolar DAC analog outputs (±10 VDC) 2 x general purpose relay contacts	4-3740-00-A002-00000
RJ-45 connectors 2 x 16-bit bipolar ADC analog inputs (±10 VDC) 2 x 12-bit bipolar DAC analog outputs (±10 VDC) 2 x general purpose relay contacts	4-3740-00-C002-00000

Note

Revisions 101 and older of the ACC-65M could only support the 12-bit ADC inputs which allowed the user to have ± 2047 counts of resolution. The 16-bit ADCs provide ± 32767 counts.

Environmental Specifications

Description	Specification	Notes
Operating Temperature	0 °C to 50 °C	
Storage Temperature	-25 °C to 70 °C	
Humidity	5% to 95%	Non-Condensing Relative Humidity

Electrical Specifications

Logic Power	
Required Voltage	24 V_{DC}
Current Requirements	1.5 A
Permitted Time at Peak Current	2 seconds

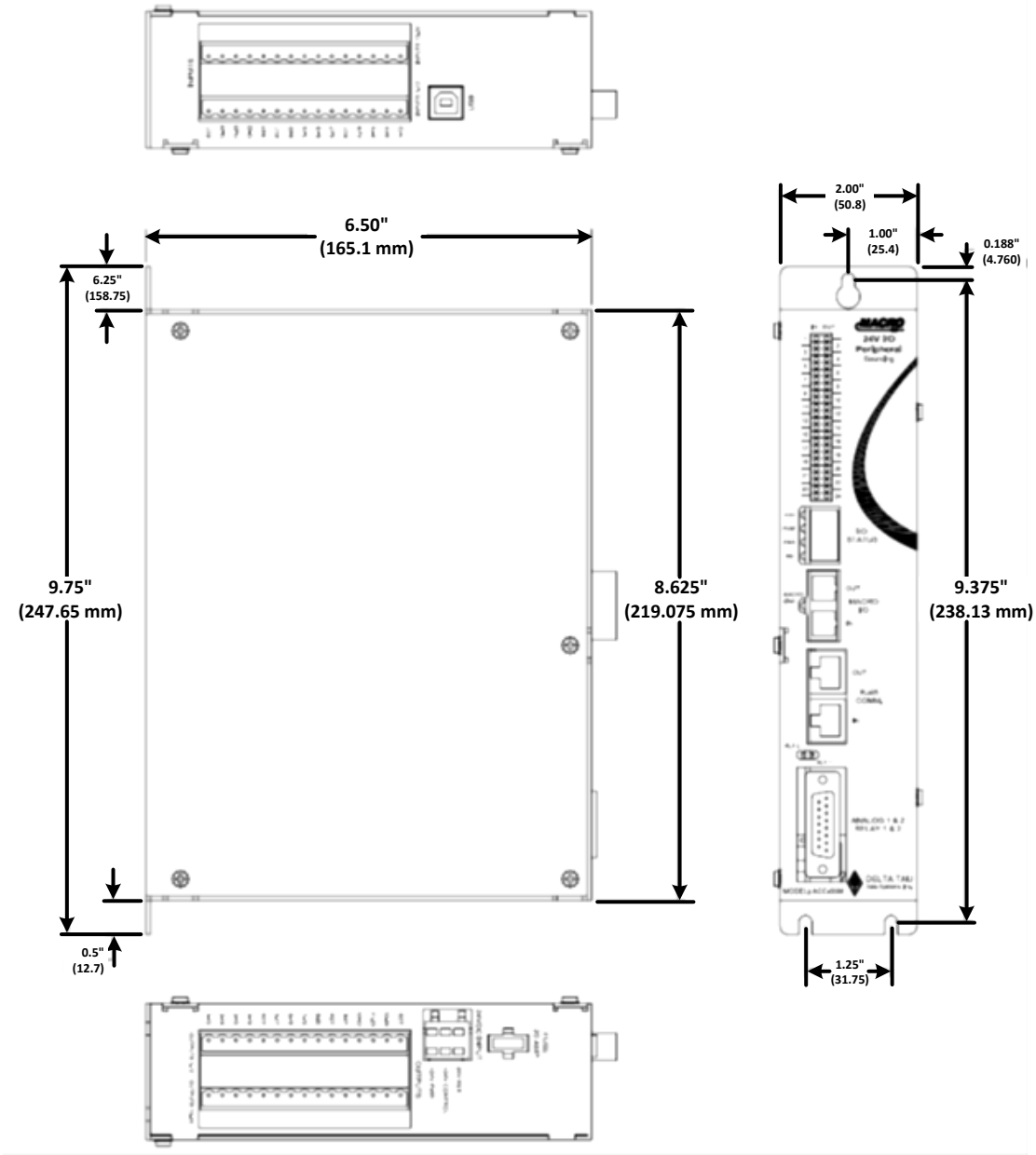
Digital Inputs	
Voltage Range	12 – 24 V_{DC}
Continuous Current Rating	1 Amp per channel
Peak Current Rating	2 Amps per channel
Permitted Time at Peak Current	2 seconds
Direction	Sourcing or Sinking (see wiring samples)

Digital Outputs	
Voltage Range	0 – 24 V_{DC}
Continuous Current Rating	600 mA per channel
Peak Current Rating	1.2 Amps per channel
Permitted Time at Peak Current	2 seconds

Analog Inputs	
Maximum Input Voltage Range	± 10 V
Resolution	16 bits
16-bit ADC Chip	Burr Brown ADS8361E
12-bit ADC Chip (Rev 1 and older)	Burr Brown ADS7861E

Analog Outputs	
Maximum Output Voltage Range	± 10 V
Output Polarity	Bipolar
Resolution	12 bits
DAC Type	Filtered PWM

Physical layout, Mounting



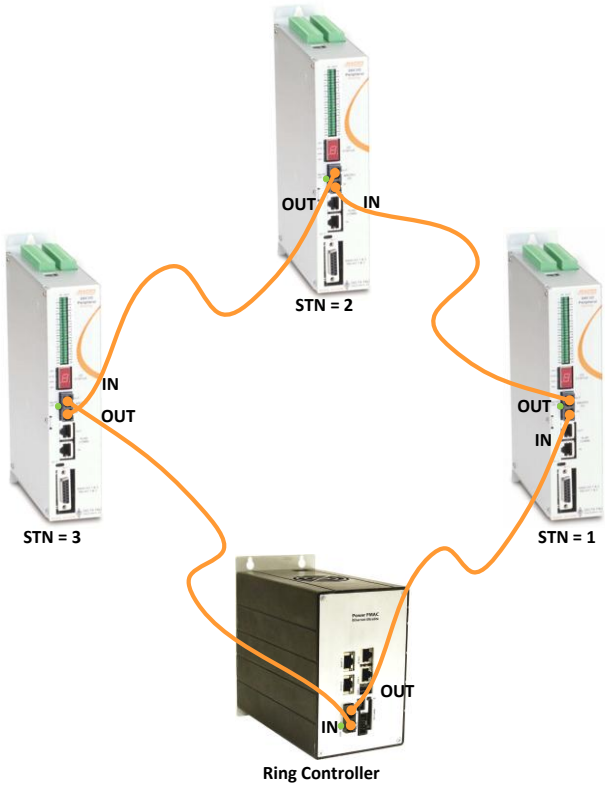
USING THE ACC-65M WITH POWER PMAC3

A Power PMAC3 Style MACRO Ring Controller can be one of the following hardware:

- Power UMAC with ACC-5E3
- Power EtherLite
- Power Brick (equipped with MACRO)
  Power Brick AC, Power Brick LV, Power Brick Controller

The first step into setting up the ACC-65M is to make sure that the MACRO cables are plugged-in in the correct manner. The OUT from the Ring Controller or previous device goes into the IN of the ACC-65M. The IN of the ACC-65M goes into the OUT of the ring controller or the next device on the ring.

For example, the illustration below shows how a MACRO ring with three ACC-65Ms is typically connected:



Note

The MACRO link LED must be green on all the devices in the MACRO ring for the software setup to work properly.

Step 1: Preparing the Ring Controller

The Power PMAC used to control a MACRO ring must be configured as a ring controller in order to establish communication and transfer data over the ring.

Following, is a summary list of the relevant parameters which need to be set properly on the Ring Controller side to allow proper functionality of the MACRO ring, and configuration of the ACC-65M.

Structure Element	Typical Setting
Sys.ClockSource (Set by Firmware)	48
Gate3[*i*].PhaseFreq	9000
Gate3[*i*].ServoClockDiv	3
Sys.ServoPeriod = 1000*(Gate3[*i*].ServoClockDiv+1)/Gate3[*i*].PhaseFreq	0.442
Sys.PhaseOverServoPeriod = 1/(Gate3[*i*].ServoClockDiv+1)	0.250
Sys.RtIntPeriod	0
Macro.TestPeriod	20
Macro.TestMaxErrors = Macro.TestPeriod / 10	2
Macro.TestReqdSynchs = Macro.TestPeriod – Macro.TestMaxErrors	18
Gate3[*i*].MacroModeA	$403000
Gate3[*i*].MacroModeB	$1000
Gate3[*i*].MacroEnableA	$*i*FC00000
Gate3[*i*].MacroEnableB	$*(i+1)*F800000

Note

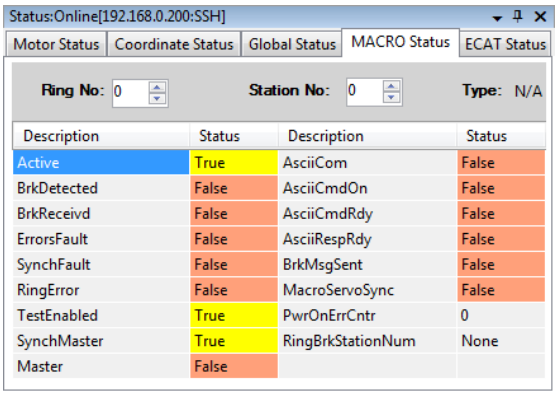
The Power PMAC can interface to up to 16 PMAC3 Style MACRO ICs.

Detailed description of these parameters can be found in the pertaining Ring Controller Hardware Reference/User manual or in the Power SRM (Software Reference Manual).

Note

These settings require a **SAVE** followed by a reset **$$$** to take effect.

Once implemented, these settings should ensure that the Power PMAC is now a MACRO ring Controller. And the MACRO Status window in the Power PMAC IDE software should look like:



Step 2: MACRO ASCII Communication

There are two possible MACRO communication methods between the ring controller and the ACC-65M:

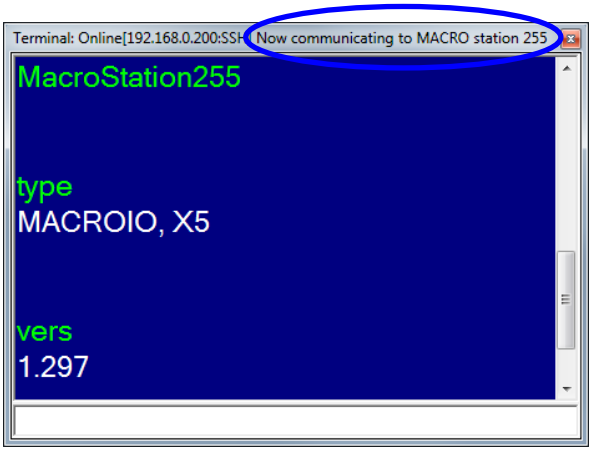
- MACRO ASCII communication
  Direct communication to the ACC-65M; it is useful for initial setup, troubleshooting, and allows to eventually establish Master Slave (MS) communication.
- Master Slave (MS) communication
  Establishing MS commands (through an I/O node) is ultimately what we want.

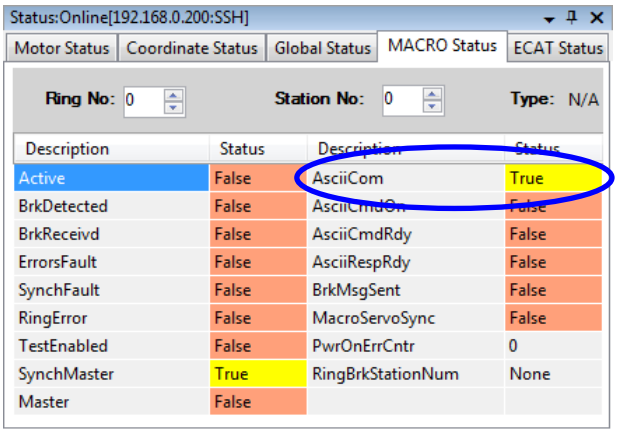
Note	Make sure that the watch window does not contain any MS{} commands prior to establishing Master Slave communication. This will latch a MACRO communication error (MACRO Status window).

Note	If the ACC-65M is to be inserted into an existing MACRO ring system. It may be more practical to place it in a MACRO ring all by itself with the ring controller. Set up and save all the necessary parameters, and then place it back into the system with the other devices.

Note	If the ACC-65M has been initialized and set up previously then it may have a station number saved to it. If you know that number (e.g. I11=1), then you would address it with the command **MacroStation1**.

If the ACC65M is at factory default settings then the user needs to issue a **MacroStation255**. This command searches the MACRO ring for new and unassigned devices. If successful, the AsciiCom status bit is highlighted in the MACRO status window:

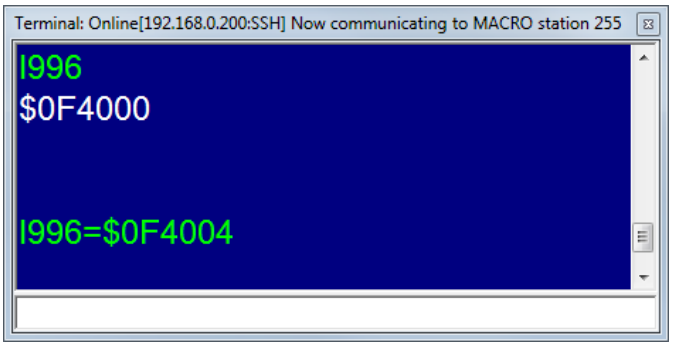




Now, you are talking directly to the ACC-65M. You should be able to issue commands such as type **TYPE**, version **VERS** etc…

The goal of MACRO ASCII communication is to enable a selected I/O node over which Master Slave communication can then be used to set up the rest of the necessary parameters of the ACC-65M.

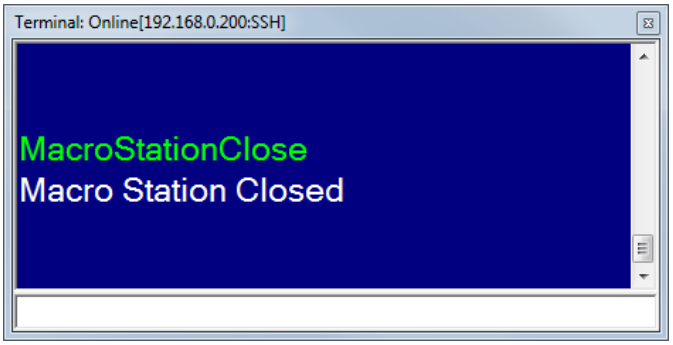
Choosing I/O node #2 as an example, enabling it is done through **I996**:



Note

One I/O node is sufficient for transferring all the data available on the ACC-65M.

Issue a **MacroStationClose** to terminate MACRO ASCII communication:



Step 3: Finishing up the ACC-65M Setup

Having enabled a selected I/O node on ACC-65M (i.e. node 2), the corresponding I/O node should be enabled on the ring controller side. For example, at MACRO IC 0, Bank A, node 2:

Gate3[0].MacroEnableA = Gate3[0].MacroEnableA | $400

Master Slave communication should be now available over I/O node 2. And the following parameters can be downloaded from the project editor. For example, station number 1 and I/O node 2:

```
MS2,I11=1               // Station number assignment (user configurable) for future
                        // MACRO ASCII communication (e.g. MACSTA1)

MS2,I992=6527           // See euqation below
MS2,I997=0              // See equation below

MS2,I995=$4080          // Typical setting for MACRO slave device
MS2,I996=$0F8004        // Nodes enabling, e.g. I/O node #2

MS2,I8=181              // Ring check period (see equation below)
MS2,I9=28               // Maximum ring error count  (see equation below)
MS2,I10=153             // Minimum synch packet count  (see equation below)
```

MS{}, I992, and **I997** are set so that the phase frequency is the same as the ring controller:

$$MS2, I992 = Ceil\left(\left(\frac{117964.8 \times 1000}{Gate3[i].PhaseFreq} - 3\right) \div 2\right)$$ // Where ceil is rounding to the higher integer

$$MS2, I997 = Gate3[i].PhaseClockDiv$$

If the clock settings are not at default, **MS{},I8, I9**, and **I10** can be calculated using the following equations. Assuming a typical ring check period (RingCheckPeriod) of 20 milliseconds and a fatal packet error (MaxErrorPercent) of 15 percent:

$$MS2, I8 = INT\left(\frac{(Ringcheck\,Period \times 117964.8) \times (MS2, I997 + 1)}{(2 \times MS2, I992 + 3)} + 1\right)$$

$$MS2, I9 = INT\left(\frac{(MS2, I8 \times MaxErrorPercent)}{100} + 1\right)$$

$$MS2, I10 = MS2, I8 - MS2, I9$$

Note

These equations must be computed ahead of time, expressions cannot be written directly into MS{} variables.

These settings must be retained on the ACC-65M. This is done by issuing a save (e.g. MSSAVE2), followed by a reset (e.g. MS$$$2) to take effect:

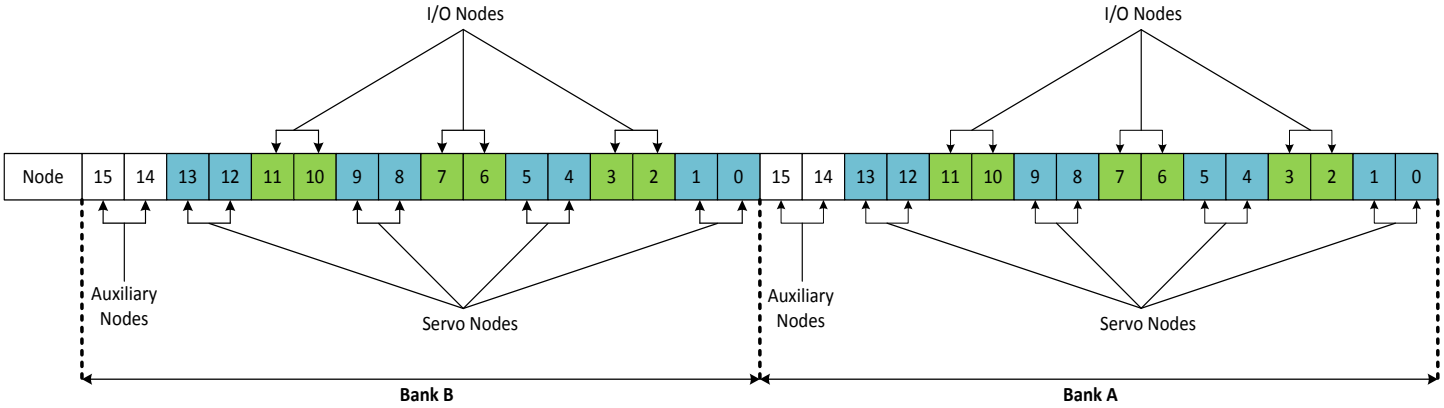
Step 4: I/O Data Registers

A single I/O node is sufficient for transferring the data to/from the ACC-65M. This is handled automatically in the firmware. The user's responsibility is choosing an available I/O node, enabling it per the example above, and finding the corresponding register or data element structure (listed in the tables below) for reading/writing to the data.

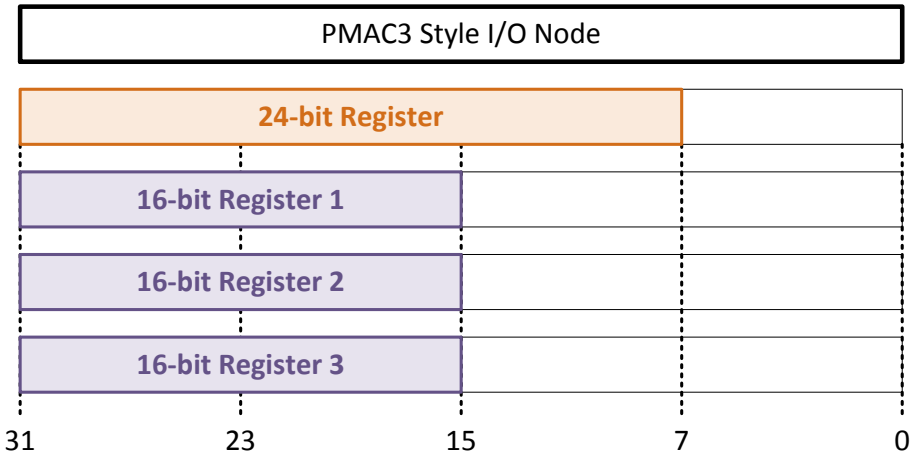
A MACRO IC consists of a number of auxiliary, servo, and I/O nodes:

- Auxiliary nodes are Master/Control registers and for internal firmware use.
- Servo nodes carry information such as feedback, commands, and flags for motor control.
- I/O nodes are by default unoccupied and are configurable for transferring miscellaneous data.

Each PMAC3 style MACRO IC consists of 32 nodes: 4 auxiliary, 16 servo, and 12 I/O nodes:


I/O Nodes
I/O Nodes
Node
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Auxiliary Nodes
Servo Nodes
Auxiliary Nodes
Servo Nodes
Bank B
Bank A

Each I/O node consists of 1 x 24-bit and 3 x 16-bit data registers residing in the following fields:



Note

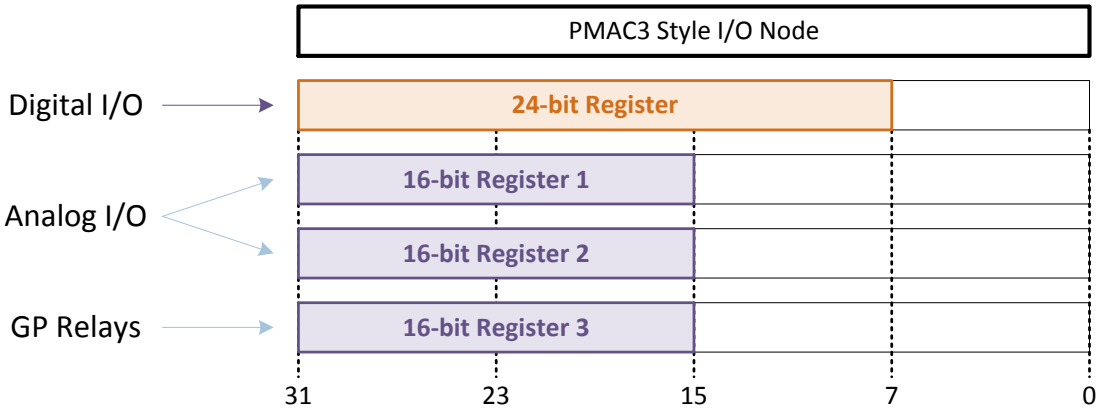
The Power PMAC can interface with up to 16 PMAC3 Style MACRO ICs. ICs present are reported by the variable **Macro.IC3s.**

Step 5: Using the ACC-65M Data

Having configured the following:

- Set up the MACRO ring controller
- Set up the phase clock to be the same across the ring
- Enabled a selected I/O node on the ACC-65M
- Enabled the corresponding I/O node on the ring controller side
- Saved and reset both the ACC-65M and the ring controller

The ACC-65M data should now be available to access from the ring controller side. The ACC-65M firmware places the data automatically in the following data registers of a selected I/O node:



And each I/O node possesses data structure elements for inputs and outputs separately for either bank:

Bank B		Bank A		Data Register
Inputs	**Outputs**	**Inputs**	**Outputs**	
Gate3[*i*].MacroInB[*j*][0]	Gate3[*i*].MacroOutB[*j*][0]	Gate3[*i*].MacroInA[*j*][0]	Gate3[*i*].MacroOutA[*j*][0]	24-bit
Gate3[*i*].MacroInB[*j*][1]	Gate3[*i*].MacroOutB[*j*][1]	Gate3[*i*].MacroInA[*j*][1]	Gate3[*i*].MacroOutA[*j*][1]	1st 16-bit
Gate3[*i*].MacroInB[*j*][2]	Gate3[*i*].MacroOutB[*j*][2]	Gate3[*i*].MacroInA[*j*][2]	Gate3[*i*].MacroOutA[*j*][2]	2nd 16-bit
Gate3[*i*].MacroInB[*j*][3]	Gate3[*i*].MacroOutB[*j*][3]	Gate3[*i*].MacroInA[*j*][3]	Gate3[*i*].MacroOutA[*j*][3]	3rd 16-bit

Where:
- *i* is the PMAC3 Style MACRO IC index
- *j* is the I/O node number.

Note

Bitwise mapping, and signed assignments into the PMAC3 Style MACRO structure elements require Power PMAC firmware version **1.5.8.305** or newer.

Note

Power PMAC firmware versions older than **1.5.8.305** must use explicit address offsets found in the memory map appendix section.

Below, are example tables showing I/O Node numbers of the first 4 PMAC3 Style MACRO ICs:

	Gate3[0]											
	Bank A						Bank B					
ACC-65M I/O Node#	2	3	6	7	10	11	2	3	6	7	11	12
Ring Controller I/O Node **[*j*]**	**2**	**3**	**6**	**7**	**10**	**11**	**18**	**19**	**22**	**23**	**26**	**27**

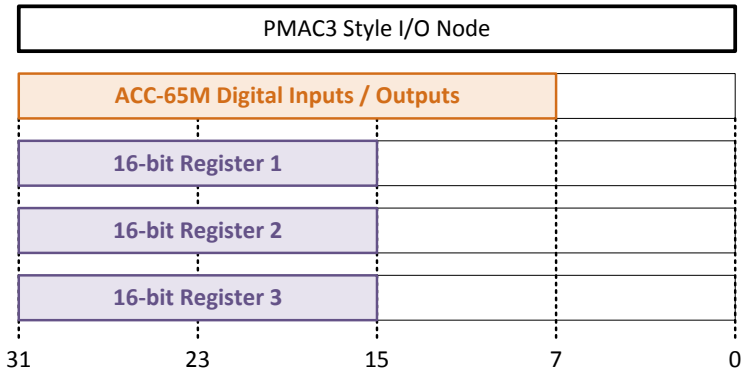
	Gate3[1]											
	Bank A						Bank B					
ACC-65M I/O Node#	2	3	6	7	10	11	2	3	6	7	11	12
Ring Controller I/O Node **[*j*]**	**34**	**35**	**38**	**39**	**42**	**43**	**50**	**51**	**54**	**55**	**58**	**59**

	Gate3[2]											
	Bank A						Bank B					
ACC-65M I/O Node#	2	3	6	7	10	11	2	3	6	7	11	12
Ring Controller I/O Node **[*j*]**	**66**	**67**	**70**	**71**	**74**	**75**	**82**	**83**	**86**	**87**	**90**	**91**

	Gate3[3]											
	Bank A						Bank B					
ACC-65M I/O Node#	2	3	6	7	10	11	2	3	6	7	11	12
Ring Controller I/O Node **[*j*]**	**98**	**99**	**102**	**103**	**106**	**107**	**114**	**115**	**118**	**119**	**122**	**123**

Digital Inputs and Outputs

The ACC-65M firmware transfers automatically the digitals inputs and outputs into/from the upper 24 bits of the 24-bit data register of the chosen I/O node.



Bank B		Bank A		Data Register
Inputs	**Outputs**	**Inputs**	**Outputs**	
Gate3[*i*].MacroInB[*j*][0]	Gate3[*i*].MacroOutB[*j*][0]	Gate3[*i*].MacroInA[*j*][0]	Gate3[*i*].MacroOutA[*j*][0]	24-bit

Where: ***i*** is the card index, and ***j*** is the I/O node number

Example: Digital I/O mapping at MACRO IC **0**, Bank **A**, I/O node **2**

Digital Outputs Bitwise

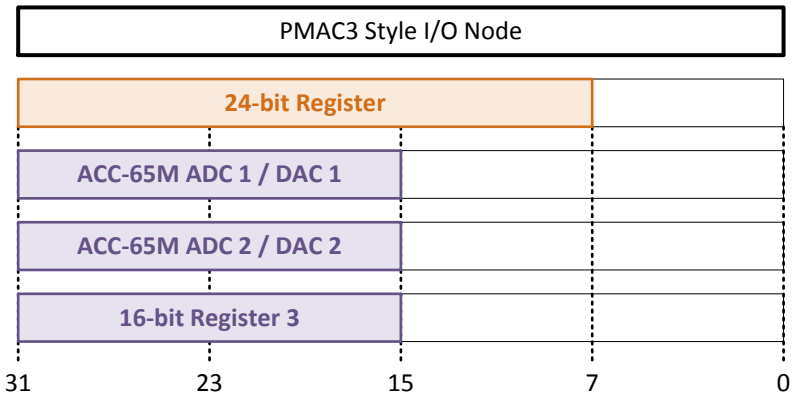
```
PTR Output1->Gate3[0].MacroOutA[2][0].8.1;
PTR Output2->Gate3[0].MacroOutA[2][0].9.1;
PTR Output3->Gate3[0].MacroOutA[2][0].10.1;
PTR Output4->Gate3[0].MacroOutA[2][0].11.1;
PTR Output5->Gate3[0].MacroOutA[2][0].12.1;
PTR Output6->Gate3[0].MacroOutA[2][0].13.1;
PTR Output7->Gate3[0].MacroOutA[2][0].14.1;
PTR Output8->Gate3[0].MacroOutA[2][0].15.1;
PTR Output9->Gate3[0].MacroOutA[2][0].16.1;
PTR Output10->Gate3[0].MacroOutA[2][0].17.1;
PTR Output11->Gate3[0].MacroOutA[2][0].18.1;
PTR Output12->Gate3[0].MacroOutA[2][0].19.1;
PTR Output13->Gate3[0].MacroOutA[2][0].20.1;
PTR Output14->Gate3[0].MacroOutA[2][0].21.1;
PTR Output15->Gate3[0].MacroOutA[2][0].22.1;
PTR Output16->Gate3[0].MacroOutA[2][0].23.1;
PTR Output17->Gate3[0].MacroOutA[2][0].24.1;
PTR Output18->Gate3[0].MacroOutA[2][0].25.1;
PTR Output19->Gate3[0].MacroOutA[2][0].26.1;
PTR Output20->Gate3[0].MacroOutA[2][0].27.1;
PTR Output21->Gate3[0].MacroOutA[2][0].28.1;
PTR Output22->Gate3[0].MacroOutA[2][0].29.1;
PTR Output23->Gate3[0].MacroOutA[2][0].30.1;
PTR Output24->Gate3[0].MacroOutA[2][0].31.1;
```

Digital Inputs Bitwise

```
PTR Input1->Gate3[0].MacroInA[2][0].8.1;
PTR Input2->Gate3[0].MacroInA[2][0].9.1;
PTR Input3->Gate3[0].MacroInA[2][0].10.1;
PTR Input4->Gate3[0].MacroInA[2][0].11.1;
PTR Input5->Gate3[0].MacroInA[2][0].12.1;
PTR Input6->Gate3[0].MacroInA[2][0].13.1;
PTR Input7->Gate3[0].MacroInA[2][0].14.1;
PTR Input8->Gate3[0].MacroInA[2][0].15.1;
PTR Input9->Gate3[0].MacroInA[2][0].16.1;
PTR Input10->Gate3[0].MacroInA[2][0].17.1;
PTR Input11->Gate3[0].MacroInA[2][0].18.1;
PTR Input12->Gate3[0].MacroInA[2][0].19.1;
PTR Input13->Gate3[0].MacroInA[2][0].20.1;
PTR Input14->Gate3[0].MacroInA[2][0].21.1;
PTR Input15->Gate3[0].MacroInA[2][0].22.1;
PTR Input16->Gate3[0].MacroInA[2][0].23.1;
PTR Input17->Gate3[0].MacroInA[2][0].24.1;
PTR Input18->Gate3[0].MacroInA[2][0].25.1;
PTR Input19->Gate3[0].MacroInA[2][0].26.1;
PTR Input20->Gate3[0].MacroInA[2][0].27.1;
PTR Input21->Gate3[0].MacroInA[2][0].28.1;
PTR Input22->Gate3[0].MacroInA[2][0].29.1;
PTR Input23->Gate3[0].MacroInA[2][0].30.1;
PTR Input24->Gate3[0].MacroInA[2][0].31.1;
```

Analog Inputs (ADCs) and Outputs (DACs)

The ACC-65M firmware transfers automatically the analog inputs and outputs from/to upper 16 bits of the 1st and 2nd 16-bit data registers of the chosen I/O node.



Bank B		Bank A		Data Register
Inputs	**Outputs**	**Inputs**	**Outputs**	
Gate3[*i*].MacroInB[*j*][1]	Gate3[*i*].MacroOutB[*j*][1]	Gate3[*i*].MacroInA[*j*][1]	Gate3[*i*].MacroOutA[*j*][1]	1st 16-bit
Gate3[*i*].MacroInB[*j*][2]	Gate3[*i*].MacroOutB[*j*][2]	Gate3[*i*].MacroInA[*j*][2]	Gate3[*i*].MacroOutA[*j*][2]	2nd 16-bit

Where: ***i*** is the card index, and ***j*** is the I/O node number

Example: Analog Input ADCs and output DACs mapping at MACRO IC **0**, Bank **A**, I/O node **2**

```
PTR ADC1->Gate3[0].MacroInA[2][1].16.16S;     // ADC #1
PTR ADC2->Gate3[0].MacroInA[2][2].16.16S;     // ADC #2

PTR DAC1->Gate3[0].MacroOutA[2][1].16.16S;    // DAC #1
PTR DAC2->Gate3[0].MacroOutA[2][2].16.16S;    // DAC #1
```

Note

The ADCs on older revisions of the ACC-65M (2-pin Molex logic connector) are 12 bits. The suffix of the address mapping should be .12.12S

Note

Typically, the ACC-65M is configured (by the factory) for unsigned data. Occasionally, it is ordered in the unsigned data format. Remove the S in the suffix for proper "unsigned" addressing.

Testing the Analog Inputs

Applying a voltage into the physical input pins, and reading the above referenced pointers for unsigned (unipolar) or signed (bipolar) data, the user should see the following.

With the 16-bit ADCs:

		Single-Ended Signal [VDC]	Differential Signal [VDC]	Software Counts
	Bipolar	-10	-5	-32768
Unipolar		0	0	0
		10	5	32768

With the 12-bit ADCs:

		Single-Ended Signal [VDC]	Differential Signal [VDC]	Software Counts
	Bipolar	-10	-5	-2048
Unipolar		0	0	0
		10	5	2048

Testing the Analog Outputs

These are ±10V outputs, where 10 volts corresponds to the value of **MS2,I992**. Remember that this is dictated by the ring phase clock, do not attempt to change it in this section.

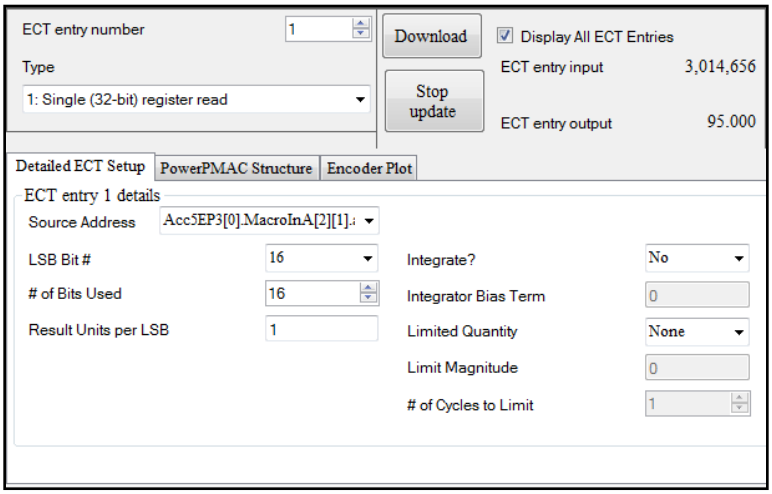
For example, with the default clock setting (e.g. MS2,I992=6527) and by writing to the analog output data register or suggested pointer, the user should see:

Pointer	Single Ended [VDC]	Differential [VDC]
-6527	-10	-20
-3264	-5	-10
0	0	0
3264	+5	+10
6527	+10	+20

Using the ADCs for Servo Feedback

Using an analog ADC input for servo requires bringing it into the encoder conversion table (ECT). Using the automatic ECT utility in the IDE software:

- Type: Single 32-bit register read
- Source Address: I/O node structure element address (i.e. Gate3[*i*].MacroInA[*j*][1])
- LSB Bit#: starting bit of ADC data (typically 16)
- #of Bits Used: ADC data number of bits (16 or 12)
- Result Units: set to 1 to shift data 16 bits for proper scaling



Alternately, using the ECT structure elements:

```
EncTable[1].type = 1
EncTable[1].pEnc = Gate3[0].MacroInA[2][1].a
EncTable[1].index1 = 16
EncTable[1].index2 = 16
EncTable[1].index3 = 0
EncTable[1].index4 = 0
EncTable[1].index5 = 0
EncTable[1].ScaleFactor = 1 / EXP2(16)
```

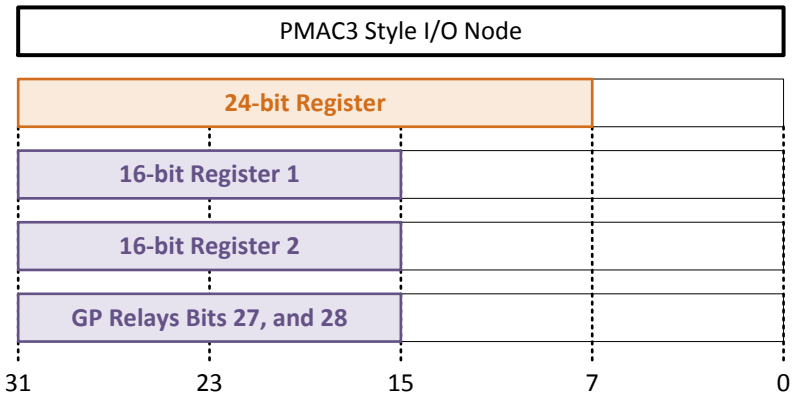
The ADC data is now processed in the encoder conversion table.

A motor element structure can point to it.
Example: Motor[1].pMasterEnc = EncTable[1].a

Or it can be accessed manually using a pointer. Note that you would need to multiply by the scale factor (or divide by 2^16 in this example) for proper scaling.
Example: PTR ECT1Result->EncTable[1].PrevEnc

General Purpose Relay Outputs

The ACC-65M firmware transfers automatically the general purpose relay outputs 1 and 2 into bits 27 and 28 respectively of the 3rd 16-bit I/O data register.



Bank B		Bank A		Data Register
Inputs	**Outputs**	**Inputs**	**Outputs**	
Gate3[*i*].MacroInB[*j*][3]	Gate3[*i*].MacroOutB[*j*][3]	Gate3[*i*].MacroInA[*j*][3]	Gate3[*i*].MacroOutA[*j*][3]	3rd 16-bit

Example: General purpose relay outputs mapping at MACRO IC **0**, both banks, and all nodes:

	I/O	GP Relay #1	GP Relay #2
Bank A	**2**	PTR GpRelay1->Gate3[0].MacroOutA[**2**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutA[**2**][3].28.1
	3	PTR GpRelay1->Gate3[0].MacroOutA[**3**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutA[**3**][3].28.1
	6	PTR GpRelay1->Gate3[0].MacroOutA[6][3].27.1	PTR GpRelay2->Gate3[0].MacroOutA[**6**][3].28.1
	7	PTR GpRelay1->Gate3[0].MacroOutA[**7**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutA[**7**][3].28.1
	10	PTR GpRelay1->Gate3[0].MacroOutA[**10**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutA[**10**][3].28.1
	11	PTR GpRelay1->Gate3[0].MacroOutA[11][3].27.1	PTR GpRelay2->Gate3[0].MacroOutA[**11**][3].28.1
Bank B	**2**	PTR GpRelay1->Gate3[0].MacroOutB[**2**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutB[**2**][3].28.1
	3	PTR GpRelay1->Gate3[0].MacroOutB[**3**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutB[**3**][3].28.1
	6	PTR GpRelay1->Gate3[0].MacroOutB[**6**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutB[**6**][3].28.1
	7	PTR GpRelay1->Gate3[0].MacroOutB[**7**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutb[**7**][3].28.1
	10	PTR GpRelay1->Gate3[0].MacroOutB[**10**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutB[**10**][3].28.1
	11	PTR GpRelay1->Gate3[0].MacroOutB[**11**][3].27.1	PTR GpRelay2->Gate3[0].MacroOutB[**11**][3].28.1

Testing the General Purpose Relays

The following table summarizes the relay functions. That is the relationship between the common line and the normally open / normally closed lines:

GP Relay 1	Connection between pins #13 (COM) and #14 (NO)	Connection between pins #13 (COM) and #6 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

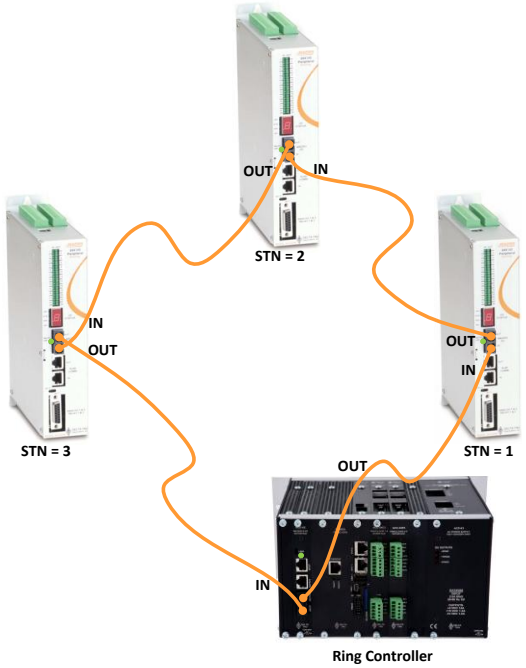
GP Relay 2	Connection between pins #7 (COM) and #8 (NO)	Connection between pins #7 (COM) and #15 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

USING THE ACC-65M WITH POWER PMAC2

A Power PMAC2 Style MACRO Ring Controller is comprised of a Power UMAC with one or more ACC-5Es in the rack.

The first step into setting up the ACC-65M is to make sure that the MACRO cables are plugged-in in the correct manner. The OUT from the Ring Controller or previous device goes into the IN of the ACC-65M. The IN of the ACC-65M goes into the OUT of the ring controller or the next device on the ring.

For example, the illustration below shows how a MACRO ring with three ACC-65Ms is typically connected:



Note

The MACRO link LED must be green on all the devices in the MACRO ring for the software setup to work properly.

Note

The Power UMAC with ACC-5E is the only configuration in which a Power PMAC interfaces to a PMAC2 Style MACRO IC.

Step 1: Preparing the Ring Controller

The Power PMAC used to control a MACRO ring must be configured as a ring controller in order to establish communication and transfer data over the ring.

Following, is a summary list of the relevant structure elements which need to be set properly on the Ring Controller side to allow proper functionality of the MACRO ring, and configuration of the ACC-65M:

Structure element	Typical Setting
Sys.ClockSource (Set by Firmware)	32
Gate2[*i*].PwmPeriod	6527
Gate2[*i*].PhaseClockDiv	0
Gate2[*i*].ServoClockDiv	3
Sys.ServoPeriod=(2*Gate2[*i*].PwmPeriod+3)*(Gate2[*i*].PhaseClockDiv+1)*(Gate2[*i*].ServoClockDiv+1)/117964.8	0.442
Sys.PhaseOverServoPeriod=1/(Gate2[*i*].ServoClockDiv+1)	0.250
Sys.RtIntPeriod	0
Macro.TestPeriod	20
Macro.TestMaxErrors = Macro.TestPeriod / 10	2
Gate2[*i*].MacroMode	$4030
Gate2[*i*].MacroEnable	$*i*FC000

Where *i* is the ACC-5E (Gate2[*i*]) index.

Detailed description of these parameters can be found in the pertaining Ring Controller Hardware Reference/User manual or in the Power SRM (Software Reference Manual).

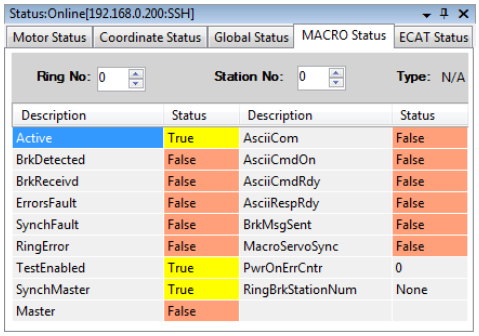
Note

The Power PMAC can interface with up to 32 PMAC2 Style MACRO ICs or up to 8 fully populated ACC-5Es.

Note

These settings require a **SAVE** followed by a reset **$$$** to take effect.

Once implemented, these settings should ensure that the Power PMAC is now a MACRO ring Controller. And the MACRO Status window in the Power PMAC IDE software should look like:



Step 2: MACRO ASCII Communication

There are two possible MACRO communication methods between the ring controller and the ACC-65M:

- MACRO ASCII communication
  Direct communication to the ACC-65M; it is useful for initial setup, troubleshooting, and allows to eventually establish Master Slave (MS) communication.
- Master Slave (MS) communication
  Establishing MS commands (through an I/O node) is ultimately what we want.

Note

Make sure that the watch window does not contain any MS{} commands prior to establishing Master Slave communication. This will latch a MACRO communication error (MACRO Status window).

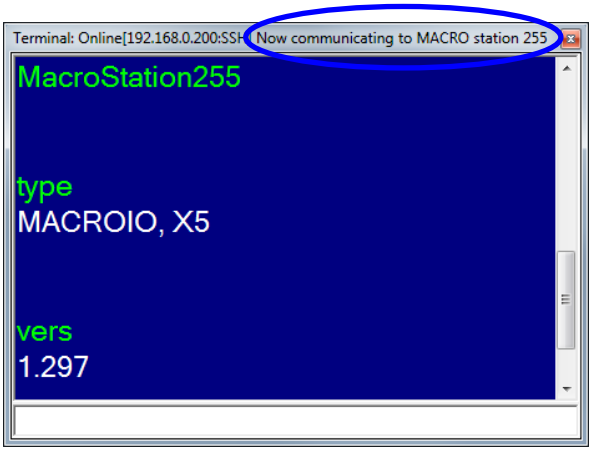
Note

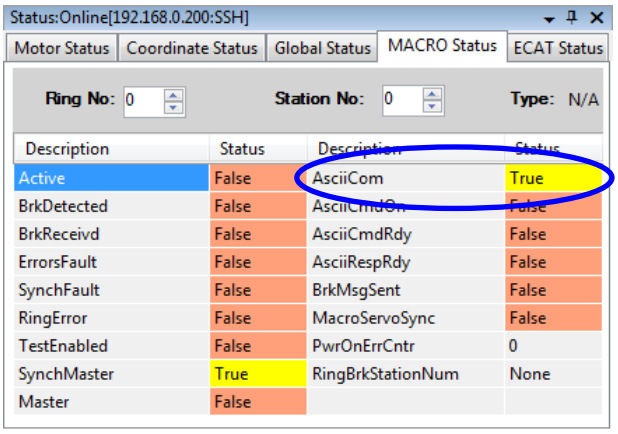
If the ACC-65M is to be inserted into an existing MACRO ring system. It may be more practical to place it in a MACRO ring all by itself with the ring controller. Set up and save all the necessary parameters, and then place it back into the system with the other devices.

Note

If the ACC-65M has been initialized and set up previously then it may have a station number saved to it. If you know that number (e.g. I11=1), then you would address it with the command **MacroStation1**.

If the ACC65M is at factory default settings then the user needs to issue a **MacroStation255**. This command searches the MACRO ring for new and unassigned devices. If successful, the AsciiCom status bit is highlighted in the MACRO status window:

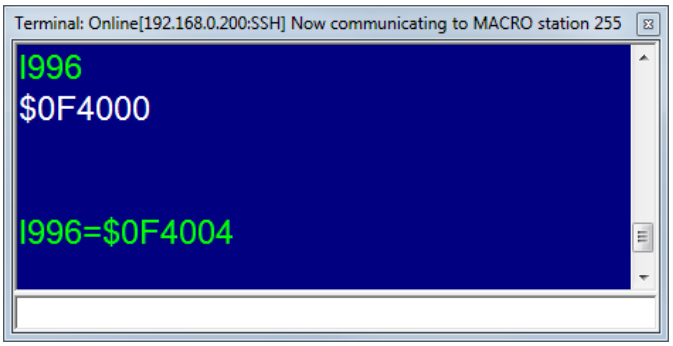




Now, you are talking directly to the ACC-65M. You should be able to issue commands such as type **TYPE**, version **VERS** etc…

The goal of MACRO ASCII communication is to enable a selected I/O node over which Master Slave communication can be used to set up the rest of the necessary parameters of the ACC-65M.

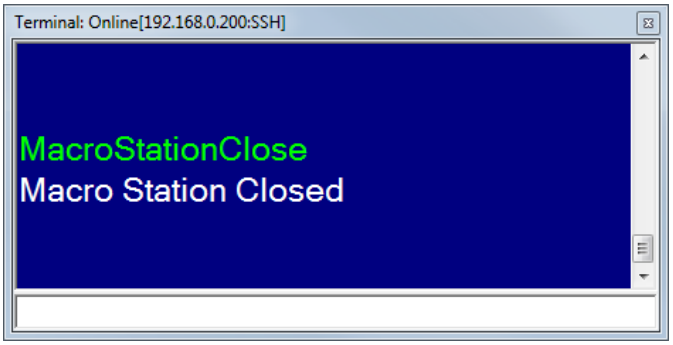
Choosing I/O node #2 as an example, enabling it is done through **I996**:



Note

One I/O node is sufficient for transferring all the data available on the ACC-65M.

Issue a **MacroStationClose** to terminate MACRO ASCII communication:



Step 3: Finishing up the ACC-65M Setup

Having enabled a selected I/O node on ACC-65M through MACRO ASCII (i.e. node 2), the corresponding I/O node should be enabled on the ring controller side. For example at MACRO IC 0:

Gate2[0].MacroEnable = Gate2[0].MacroEnable | $4

Master Slave communication should be now available over I/O node 2. And the following parameters can be downloaded from the project editor. For example, station number 1 and I/O node 2:

```
MS2,I11=1              // Station number assignment (user configurable) for future
                       // MACRO ASCII communication (e.g. MACSTA1)

MS2,I992=6527          // See below
MS2,I997=0             // See below

MS2,I995=$4080         // Typical setting for MACRO slave device
MS2,I996=$0F8004       // Nodes enabling, e.g. I/O node #2

MS2,I8=181             // Ring check period (see equation below)
MS2,I9=28              // Maximum ring error count  (see equation below)
MS2,I10=153            // Minimum synch packet count  (see equation below)
```

MS{}, I992, and **I997** are set so that the phase frequency is the same as the ring controller:

$$MS2, I992 = Gate2[0].PwmPeriod$$

$$MS2, I997 = Gate2[0].PhaseClockDiv$$

If the clock settings are not at default, **MS{},I8, I9**, and **I10** can be calculated using the following equations. Assuming a typical ring check period (RingCheckPeriod) of 20 milliseconds and a fatal packet error (MaxErrorPercent) of 15 percent:

$$MS2, I8 = INT\left(\frac{(Ringcheck\,Period \times 117964.8) \times (MS2, I997 + 1)}{(2 \times MS2, I992 + 3)} + 1\right)$$

$$MS2, I9 = INT\left(\frac{(MS2, I8 \times MaxErrorPercent)}{100} + 1\right)$$

$$MS2, I10 = MS2, I8 - MS2, I9$$

Note

These equations must be computed explicitly ahead of time, expressions cannot be written directly into MS{} variables.

These settings must be retained on the ACC-65M. This is done by issuing a save (e.g. MSSAVE2), followed by a reset (e.g. MS$$$2) to take effect:

```
MSSAVE2
MS$$$2
```

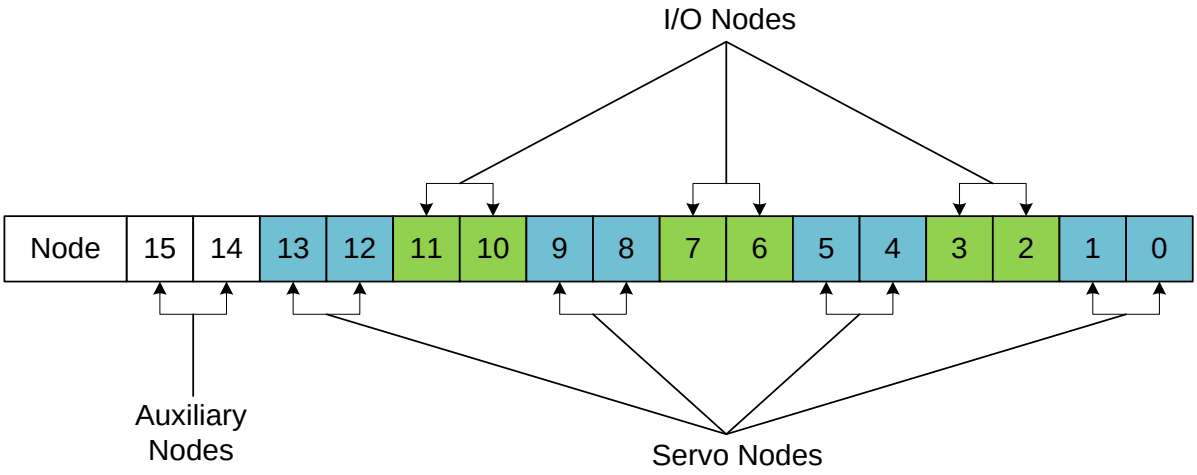
Step 4: I/O Data Registers

A single I/O node is sufficient for transferring the data to/from the ACC-65M. This is handled automatically in the firmware. The user's responsibility is choosing an available I/O node, enabling it per the example above, and finding the corresponding register or data element structure (listed in the tables below) for reading/writing to the data.

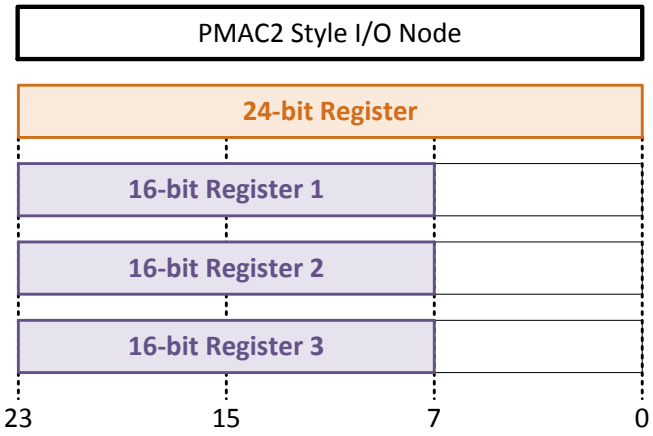
A MACRO IC consists of a number of auxiliary, servo, and I/O nodes:

- Auxiliary nodes are Master/Control registers and for internal firmware use.
- Servo nodes carry information such as feedback, commands, and flags for motor control.
- I/O nodes are by default unoccupied and are configurable for transferring miscellaneous data.

Each PMAC2 Style MACRO IC consists of 16 nodes: 2 auxiliary, 8 servo, and 6 I/O nodes:



Each I/O node register consists of one 24-bit and three 16-bit data registers placed in the following fields:



Note

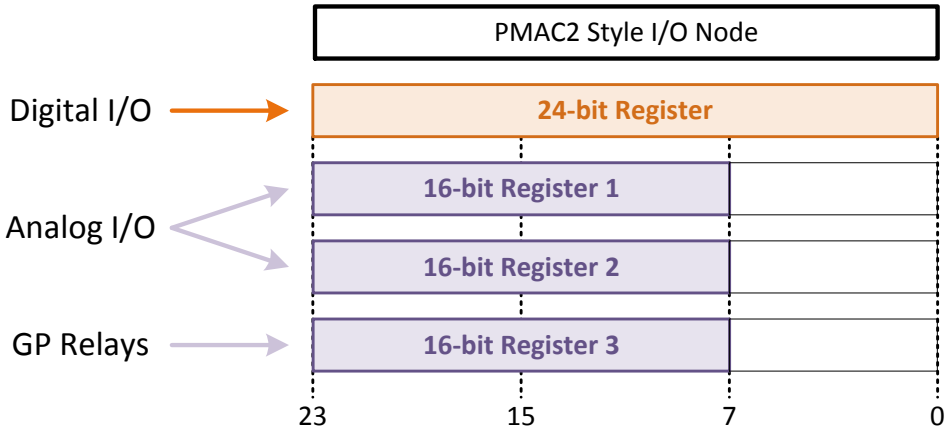
A Power PMAC CPU can interface with up to 32 PMAC2 Style MACRO ICs. ICs present are reported by the variable **Macro.ICs**

Step 5: Using the ACC-65M Data

Having configured the following:

- Set up the MACRO ring controller
- Set up the phase clock to be the same across the ring
- Enabled a selected I/O node on the ACC-65M
- Enabled the corresponding I/O node on the ring controller side
- Saved and reset both the ACC-65M and the ring controller

The ACC-65M data should now be available to access from the ring controller side at the specified I/O node with the data residing is in the following fields:



The PMAC2 Style MACRO IC structure elements for these registers are:

Structure Element	Data Register
Gate2[*i*].Macro[*j*][0]	24-bit
Gate2[*i*].Macro[*j*][0]	16-bit
Gate2[*i*].Macro[*j*][0]	16-bit
Gate2[*i*].Macro[*j*][0]	16-bit

Where:

- *i* is the PMAC2 Style MACRO IC index
- *j* is the I/O node number.

Note

Bitwise mapping, and signed assignments into the PMAC2 Style MACRO structure elements require Power PMAC firmware version **1.5.8.305** or newer.

Note

Power PMAC firmware versions older than **1.5.8.305** must use explicit address offsets found in the memory map appendix section.

Below, are example tables showing I/O Node numbers of the first 4 PMAC2 Style MACRO ICs:

	Gate2[**0**]					
ACC-65M I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node [*j*]	**2**	**3**	**6**	**7**	**10**	**11**

	Gate2[**1**]					
ACC-65M I/O Node#	2	3	6	7	11	12
Ring Controller I/O Node [*j*]	**18**	**19**	**22**	**23**	**26**	**27**

	Gate2[**2**]					
ACC-65M I/O Node#	2	3	6	7	10	11
Ring Controller I/O Node [*j*]	**34**	**35**	**38**	**39**	**42**	**43**

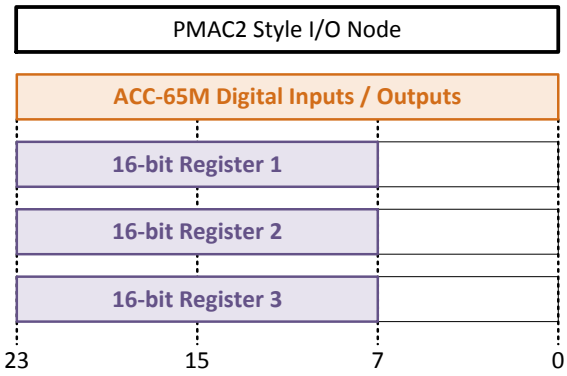
	Gate2[**3**]					
ACC-65M I/O Node#	2	3	6	7	11	12
Ring Controller I/O Node [*j*]	**50**	**51**	**54**	**55**	**58**	**59**

Note

A Power PMAC CPU can interface with up to 32 PMAC2 Style MACRO ICs. ICs present are reported by the variable **Macro.ICs**

Digital Inputs and Outputs

The ACC-65M firmware transfers automatically the digitals inputs and outputs into/from the lower 24 bit data register of the chosen I/O node.



The PMAC2 Style MACRO IC structure element for this register is:

Structure Element	Data Register
Gate2[*i*].Macro[*j*][0]	24-bit

Where:
- ***i*** is the PMAC2 Style MACRO IC index
- ***j*** is the I/O node number.

Digital Inputs

Example: Digital inputs mapping at PMAC2 MACRO IC **0**, I/O node **2**

Digital Inputs Bitwise

```
PTR Input1->Gate2[0].Macro[2][0].0.1;
PTR Input2->Gate2[0].Macro[2][0].1.1;
PTR Input3->Gate2[0].Macro[2][0].2.1;
PTR Input4->Gate2[0].Macro[2][0].3.1;
PTR Input5->Gate2[0].Macro[2][0].4.1;
PTR Input6->Gate2[0].Macro[2][0].5.1;
PTR Input7->Gate2[0].Macro[2][0].6.1;
PTR Input8->Gate2[0].Macro[2][0].7.1;
PTR Input9->Gate2[0].Macro[2][0].8.1;
PTR Input10->Gate2[0].Macro[2][0].9.1;
PTR Input11->Gate2[0].Macro[2][0].10.1;
PTR Input12->Gate2[0].Macro[2][0].11.1;
PTR Input13->Gate2[0].Macro[2][0].12.1;
PTR Input14->Gate2[0].Macro[2][0].13.1;
PTR Input15->Gate2[0].Macro[2][0].14.1;
PTR Input16->Gate2[0].Macro[2][0].15.1;
PTR Input17->Gate2[0].Macro[2][0].16.1;
PTR Input18->Gate2[0].Macro[2][0].17.1;
PTR Input19->Gate2[0].Macro[2][0].18.1;
PTR Input20->Gate2[0].Macro[2][0].19.1;
PTR Input21->Gate2[0].Macro[2][0].20.1;
PTR Input22->Gate2[0].Macro[2][0].21.1;
PTR Input23->Gate2[0].Macro[2][0].22.1;
PTR Input24->Gate2[0].Macro[2][0].23.1;
```

Note

Bitwise mapping into the PMAC2 Style MACRO structure elements require Power PMAC firmware version **1.5.8.305** or newer.

Digital Outputs

The outputs can be written to using the structure element's full word. **Example:** PMAC2 MACRO IC 0, Node 2; `PTR Outputs->Gate2[0].Macro[2][0]`

However, with the PMAC2 Style MACRO IC, the outputs require an image word to report the state of each output, and allow bitwise mapping.

This can be done in a simple PLC, and using one of the "unsigned" user shared memory data elements Sys.Udata[i]. The table to the right shows 4 of the possible 65K registers available:

Structure Element	Address Offset
Sys.Udata[4]	U.USER:16
Sys.Udata[8]	U.USER:32
Sys.Udata[12]	U.USER:48
Sys.Udata[16]	U.USER:64

Note

A large number of self-addressed (default **Sys.pushm**) pointers in Power PMAC use **Sys.Udata[0]**; therefore it is highly advised NOT to use it as a general purpose user shared memory.

Example: using Sys.Udata[4]

Digital Outputs Bitwise

```
PTR Output1->U.USER:16.0.1;
PTR Output2->U.USER:16.1.1;
PTR Output3->U.USER:16.2.1;
PTR Output4->U.USER:16.3.1;
PTR Output5->U.USER:16.4.1;
PTR Output6->U.USER:16.5.1;
PTR Output7->U.USER:16.6.1;
PTR Output8->U.USER:16.7.1;
PTR Output9->U.USER:16.8.1;
PTR Output10->U.USER:16.9.1;
PTR Output11->U.USER:16.10.1;
PTR Output12->U.USER:16.11.1;
PTR Output13->U.USER:16.12.1;
PTR Output14->U.USER:16.13.1;
PTR Output15->U.USER:16.14.1;
PTR Output16->U.USER:16.15.1;
PTR Output17->U.USER:16.16.1;
PTR Output18->U.USER:16.17.1;
PTR Output19->U.USER:16.18.1;
PTR Output20->U.USER:16.19.1;
PTR Output21->U.USER:16.20.1;
PTR Output22->U.USER:16.21.1;
PTR Output23->U.USER:16.22.1;
PTR Output24->U.USER:16.23.1;
```

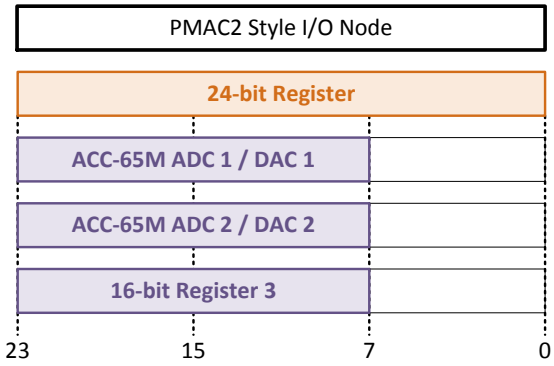
And the mirror PLC, which should be executing constantly:

```
PTR IC0_N2Twenty4->Gate2[0].Macro[2][0];     // IC 0, Node 2, 24-bit register
PTR OutputsMirror->U.USER:16;                // Sys.Udata[4], mirror word
OutputsMirror = 0;                           // Save/Initialize to zero or desired state

OPEN PLC 1
IC0_N2Twenty4 = OutputsMirror                // Update data register
CLOSE
```

Analog Inputs (ADCs) and Outputs (DACs)

The ACC-65M firmware transfers automatically the analog inputs and outputs from/to the 1st and 2nd 16-bit data registers of the chosen I/O node.



The PMAC2 Style MACRO IC structure elements for these registers are:

Structure Element	Data Register
Gate2[*i*].Macro[*j*][1]	1st 16-bit
Gate2[*i*].Macro[*j*][2]	2nd 16-bit

Where:
- ***i*** is the PMAC2 Style MACRO IC index
- ***j*** is the I/O node number.

Analog Input ADCs

The ADC inputs can be mapped directly into the node's structure elements, and read directly without further processing. Typically, the ACC-65M is configured (by the factory) for signed ADC inputs.

Example: Signed ADC inputs at PMAC2 Style MACRO IC 0, Node 2:

```
PTR ADC1->Gate2[0].Macro[2][1].8.16S  // IC 0, Node 2, ADC 1 signed
PTR ADC2->Gate2[0].Macro[2][2].8.16S  // IC 0, Node 2, ADC 2 signed
```

Example: Unsigned ADC inputs at PMAC2 Style MACRO IC 0, Node 2:

```
PTR ADC1->Gate2[0].Macro[2][1].8.16   // IC 0, Node 2, ADC 1 unsigned
PTR ADC2->Gate2[0].Macro[2][2].8.16   // IC 0, Node 2, ADC 2 unsigned
```

Note

The ADC inputs on the older revision of the ACC-65M (2-pin Molex logic connector) are 12 bits. The suffix of the address mapping should be 12.12S.

Note

Bitwise, and signed mapping into the PMAC2 Style MACRO structure elements require Power PMAC firmware version **1.5.8.305** or newer.

Analog Output DACs

The analog output DACs can be written to using the structure element's full word. **Example:** PMAC2 MACRO IC 0, Node 2, DAC 1; `PTR DAC1->Gate2[0].Macro[2][1].`

Structure Element	Address Offset
Sys.Udata[4]	U.USER:16
Sys.Udata[8]	U.USER:32
Sys.Udata[12]	U.USER:48
Sys.Udata[16]	U.USER:64

However, with the PMAC2 Style MACRO IC, the DAC outputs require an image word to report the value of each output, and allow byte-wise mapping for proper scaling.

This can be done in a simple PLC, and using one of the "unsigned" user shared memory data elements Sys.Udata[i]. The table to the right shows 4 of the possible 65K registers available:

Note

A large number of self-addressed (default **Sys.pushm**) pointers in Power PMAC use **Sys.Udata[0]**; therefore it is highly advised NOT to use it as a general purpose user shared memory.

Example: Using Sys.Udata[8] for both DACs 1, and 2

```
PTR DAC1->S.USER:32.0.16      // DAC 1, pointing to signed user shared memory
PTR DAC2->S.USER:32.8.16      // DAC 2, pointing to signed user shared memory
```

Note

Typically, the ACC-65M is configured (by the factory) for signed DAC outputs. For unsigned DACs, simply replace the S in the prefix of the assignment with a U.

And the mirror PLC, which should be executing constantly:

```
PTR N2First16->Gate[2].Macro[2][1]    // Node 2, 1st 16-bit data register
PTR N2Second16->Gate[2].Macro[2][2]   // Node 2, 2nd 16-bit data register

OPEN PLC 1
N2First16 = DAC1 * 256        // Update data register, upper 16
N2Second16 = DAC2 * 256       // Update data register, upper 16
CLOSE
```

Testing the Analog Inputs

Applying a voltage into the physical input pins, and reading the above referenced pointers for unsigned (unipolar) or signed (bipolar) data, the user should see the following.

With the 16-bit ADCs:

		Single-Ended Signal [VDC]	Differential Signal [VDC]	Software Counts
Unipolar	**Bipolar**	-10	-5	-32768
		0	0	0
		10	5	32768

With the 12-bit ADCs:

		Single-Ended Signal [VDC]	Differential Signal [VDC]	Software Counts
Unipolar	**Bipolar**	-10	-5	-2048
		0	0	0
		10	5	2048

Testing the Analog Outputs

These are ±10V outputs, where 10 volts corresponds to the value of **MS2,I992**. Remember that this is dictated by the ring phase clock, do not attempt to change it in this section.

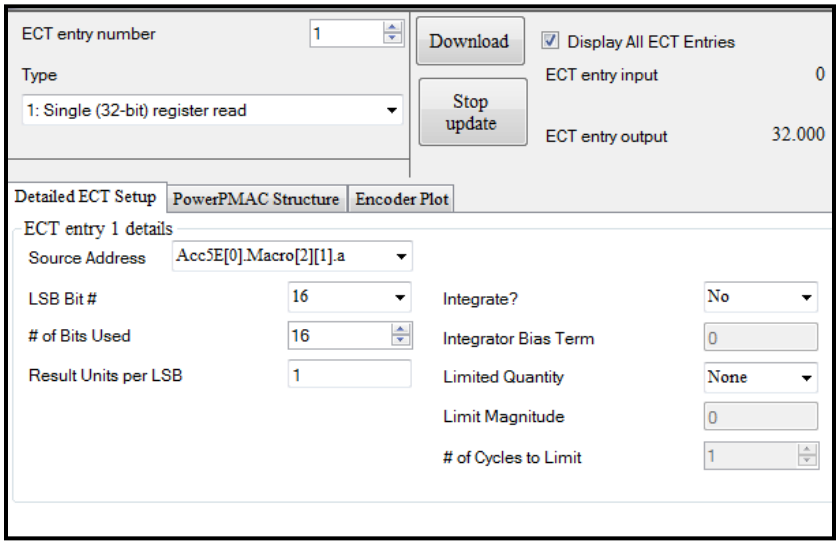
For example, with the default clock setting (e.g. MS2,I992=6527) and by writing to the analog output data register or suggested pointer, the user should see:

Pointer	Single Ended [VDC]	Differential [VDC]
-6527	-10	-20
-3264	-5	-10
0	0	0
3264	+5	+10
6527	+10	+20

Using the ADCs for Servo Feedback

Using an analog ADC input for servo requires bringing it into the encoder conversion table (ECT). Using the automatic ECT utility in the IDE software:

- Type: Single 32-bit register read
- Source Address: I/O node structure element address (i.e. Gate2[*i*].Macro[*j*][1])
- LSB Bit#: starting bit of ADC data (typically 16)
- #of Bits Used: ADC data number of bits (16 or 12)
- Result Units: set to 1 to shift data 16 bits for proper scaling



Alternately, using the ECT structure elements:

```
EncTable[1].type = 1
EncTable[1].pEnc = Gate2[0].Macro[2][1].a
EncTable[1].index1 = 16
EncTable[1].index2 = 16
EncTable[1].index3 = 0
EncTable[1].index4 = 0
EncTable[1].index5 = 0
EncTable[1].ScaleFactor = 1 / EXP2(16)
```

The ADC data is now processed in the encoder conversion table.

A motor element structure can point to it.
Example: Motor[1].pMasterEnc = EncTable[1].a

Or it can be accessed manually using a pointer. Note that you would need to multiply by the scale factor (or divide by 2^16 in this example) for proper scaling.
Example: PTR ECT1Result->EncTable[1].PrevEnc

General Purpose Relays

The general purpose relays 1 and 2 are transferred respectively into bits 19 and 20 of the 3rd 16-bit data register of the I/O node.

The PMAC2 Style MACRO IC structure element for this register is:

Structure Element	Data Register
Gate2[*i*].Macro[*j*][3]	16-bit (middle)

Where:
- ***i*** is the PMAC2 Style MACRO IC index
- ***j*** is the I/O node number.

With the PMAC2 Style MACRO IC, the GP relay outputs require an image word to report the value of each output, and allow byte-wise mapping.

This can be done in a simple PLC, and using one of the "unsigned" user shared memory data elements Sys.Udata[i]. The table to the right shows 4 of the possible 65K registers available:

Structure Element	Address Offset
Sys.Udata[4]	U.USER:16
Sys.Udata[8]	U.USER:32
Sys.Udata[12]	U.USER:48
Sys.Udata[16]	U.USER:64

Example: Using Sys.Udata[4], bits 27 and 28 respectively as mirror bits for GP Relays 1 and 2

```
PTR GPRelay1->U.USER:16.27.1; // GP Relay 1, mirror bit
PTR GPRelay2->U.USER:16.28.1; // GP Relay 2, mirror bit
```

And the mirror PLC (which should be executing constantly) for PMAC2 MACRO IC 0, node 2:

```
PTR N2Third16->Gate[2].Macro[2][2]    // Node 2, 3rd 16-bit data register

OPEN PLC 1
N2Third16 = GPRelay1 * EXP2(19)       // Update bit, place in bit #19
N2Third16 = GPRelay2 * EXP2(20)       // Update bit, place in bit #20
CLOSE
```

Testing the General Purpose Relays

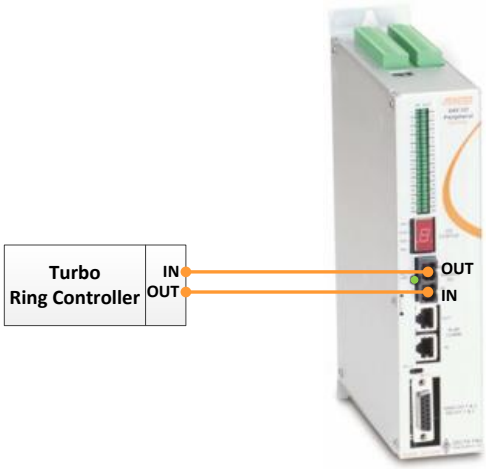
The following table summarizes the relay functions. That is the relationship between the common line and the normally open / normally closed lines:

GP Relay 1	Connection between pins #13 (COM) and #14 (NO)	Connection between pins #13 (COM) and #6 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

GP Relay 2	Connection between pins #7 (COM) and #8 (NO)	Connection between pins #7 (COM) and #15 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

USING THE ACC-65M WITH TURBO PMAC2

The first step into setting up the ACC-65M is to make sure that the MACRO cables are plugged-in in the correct manner. The OUT from the Ring Controller or previous device goes into the IN of the ACC-65M. The IN of the ACC-65M goes into the OUT of the ring controller or the next device on the ring.



A Turbo PMAC Ring Controller can be one of the following:

- Any Turbo PMAC2 Ultralite board level (e.g. PCI)
- Turbo UMAC with ACC-5E
- Turbo PMAC2 Ultralite
- Turbo Brick (equipped with MACRO)
  Geo Brick Drive, Geo Brick LV, Brick Controller

Note

The MACRO link LED must be green on all the devices in the MACRO ring for the software setup to work properly.

Step 1: Preparing the Ring Controller

The Turbo PMAC used to control the MACRO ring must be configured as a ring controller in order to establish communication and transfer data over the MACRO ring.

Following, is a summary list of the relevant parameters which need to be set properly on the Ring Controller side to allow proper functionality of the MACRO ring, and configuration of the ACC-65M.

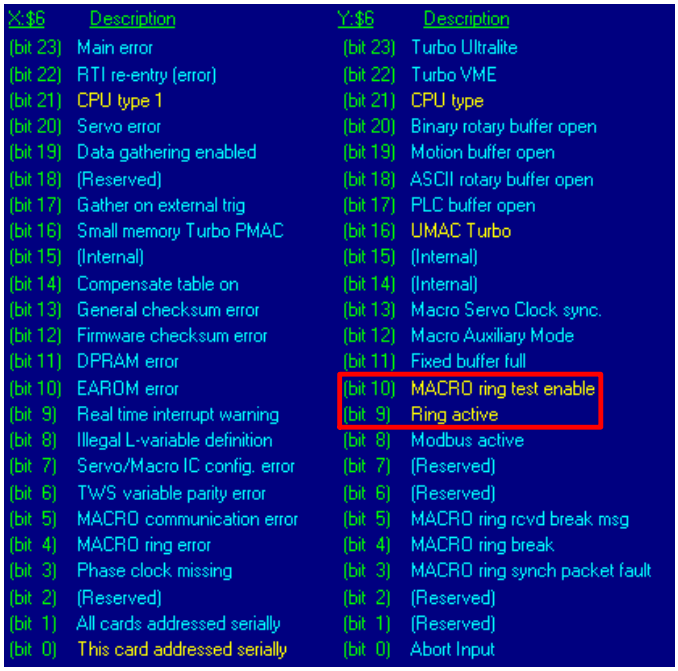
Parameter				Description	Typical Setting			
MACRO IC **0**	MACRO IC **1**	MACRO IC **2**	MACRO IC **3**					
I19				Clock Source	6807			
I6800	**I6850**	**I6900**	**I6950**	MACRO IC Max Phase	6527			
I6801	**I6851**	**I6901**	**I6951**	MACRO IC Phase Clock Divider	0			
I6802	**I6852**	**I6902**	**I6952**	MACRO IC Servo Clock Divider	3			
I8				Real time interrupt	2			
I10				Servo Interrupt Time	3713991			
I78				Enable MS, MSR, MSW commands	32			
I79				Enable MM, MMR, MMW commands	32			
I80				Ring check period	45			
I81				Maximum ring error count	2			
I82				Minimum synch packet count	13			
I6840	**I6890**	**I6940**	**I6990**	MACRO IC Ring configuration/Status	$4030	$10	$10	$10
I6841	**I6891**	**I6941**	**I6991**	MACRO IC node activation	$**0**FC000	$**1**F8000	$**2**F8000	$**3**F8000
I70	**I72**	**I74**	**I76**	MACRO IC Auxiliary register enable	$0			
I71	**I73**	**I75**	**I77**	MACRO IC Protocol node control	$0			

Detailed description of these parameters can be found in the pertaining Ring Controller Hardware Reference/User manual or in the Turbo SRM (Software Reference Manual).

Note

These settings require a SAVE followed by a reset $$$ to take effect.

Once implemented, these settings should ensure that the Turbo PMAC is now a MACRO ring Controller. And the global status in the Pewin32Pro2 software should look like:



X:$6	Description	Y:$6	Description
(bit 23)	Main error	(bit 23)	Turbo Ultralite
(bit 22)	RTI re-entry (error)	(bit 22)	Turbo VME
(bit 21)	CPU type 1	(bit 21)	CPU type
(bit 20)	Servo error	(bit 20)	Binary rotary buffer open
(bit 19)	Data gathering enabled	(bit 19)	Motion buffer open
(bit 18)	(Reserved)	(bit 18)	ASCII rotary buffer open
(bit 17)	Gather on external trig	(bit 17)	PLC buffer open
(bit 16)	Small memory Turbo PMAC	(bit 16)	UMAC Turbo
(bit 15)	(Internal)	(bit 15)	(Internal)
(bit 14)	Compensate table on	(bit 14)	(Internal)
(bit 13)	General checksum error	(bit 13)	Macro Servo Clock sync.
(bit 12)	Firmware checksum error	(bit 12)	Macro Auxiliary Mode
(bit 11)	DPRAM error	(bit 11)	Fixed buffer full
(bit 10)	EAROM error	(bit 10)	MACRO ring test enable
(bit 9)	Real time interrupt warning	(bit 9)	Ring active
(bit 8)	Illegal L-variable definition	(bit 8)	Modbus active
(bit 7)	Servo/Macro IC config. error	(bit 7)	(Reserved)
(bit 6)	TWS variable parity error	(bit 6)	(Reserved)
(bit 5)	MACRO communication error	(bit 5)	MACRO ring rcvd break msg
(bit 4)	MACRO ring error	(bit 4)	MACRO ring break
(bit 3)	Phase clock missing	(bit 3)	MACRO ring synch packet fault
(bit 2)	(Reserved)	(bit 2)	(Reserved)
(bit 1)	All cards addressed serially	(bit 1)	(Reserved)
(bit 0)	This card addressed serially	(bit 0)	Abort Input

Step 2: MACRO ASCII Communication

There are two possible MACRO communication methods between the ring controller and the ACC-65M:

- MACRO ASCII communication

Direct communication to the ACC-65M; it is useful for initial setup, troubleshooting, and allows to eventually establish Master Slave (MS) communication.

- Master Slave (MS) communication

Establishing MS commands (through an I/O node) is ultimately what we want.

Note

Make sure that the watch window does not contain any MS{} commands prior to establishing Master Slave communication. This will latch a MACRO communication error in the Global Status.

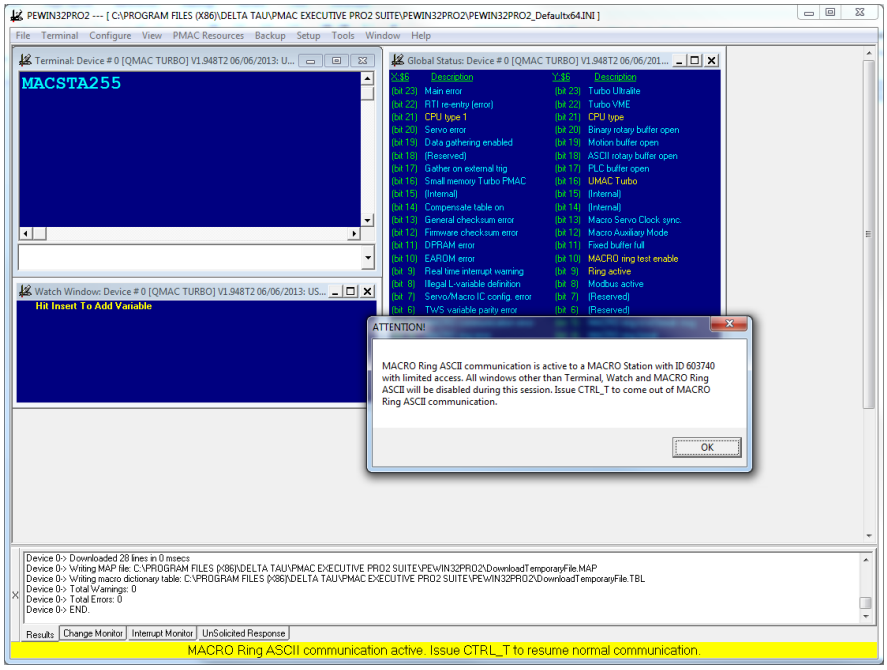
Note

If the ACC-65M is to be inserted into an existing MACRO ring system. It may be more practical to place it in a MACRO ring all by itself with the ring controller. Set up and save all the necessary parameters, and then place it back into the system with the other devices.

Note

If the ACC-65M has been initialized and set up previously then it may have a station number saved to it. If you know that number (e.g. I11=1), then you would address it with the command **MACSTA1**.

If the ACC65M is at factory default settings then the user needs to issue a **MACSTA255**. This command searches the MACRO ring for new and unassigned (station#) devices. The following message appears in the Pewin32Pro2 software, and a notification (yellow ribbon) appears in the bottom of the window indicating that MACRO ASCII communication is now active:


MACSTA255
Hit Insert To Add Variable
ATTENTION!
MACRO Ring ASCII communication is active to a MACRO Station with ID 603740 with limited access. All windows other than Terminal, Watch and MACRO Ring ASCII will be disabled during this session. Issue CTRL_T to come out of MACRO Ring ASCII communication.
OK
MACRO Ring ASCII communication active. Issue CTRL_T to resume normal communication.

Now, you are talking directly to the ACC-65M. You should be able to issue commands such as type (TYP), version (VER) etc…

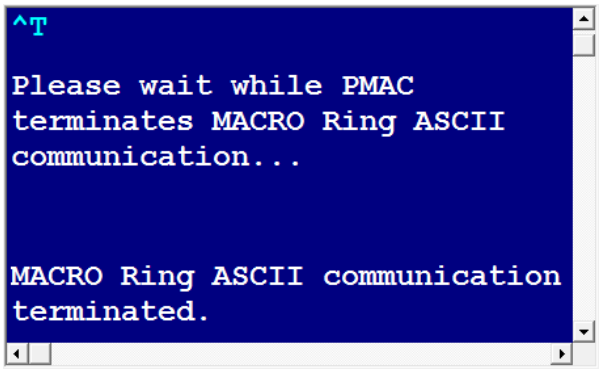
The goal of MACRO ASCII communication is to enable a selected I/O node to allow Master Slave communication from the master which then can be used to set up the rest of the necessary parameters on the ACC-65M.

Choosing I/O node #2 as an example, enabling it is done through **I996**:

Note

One I/O node is sufficient for transferring all the data available on the ACC-65M.

Press CTRL-T (^T) to exit MACRO ASCII communication:



Note

The yellow notification should now disappear. And communication is re-established with the ring controller.

Step 3: Finishing up the ACC-65M Setup

Having enabled a selected I/O node on ACC-65M, the corresponding I/O node should be enabled on the ring controller side. For example, at MACRO IC 0 node 2: **I6841=I6841|$4**.

Master Slave communication should be now available over I/O node 2. And the following parameters can be downloaded from the editor window:

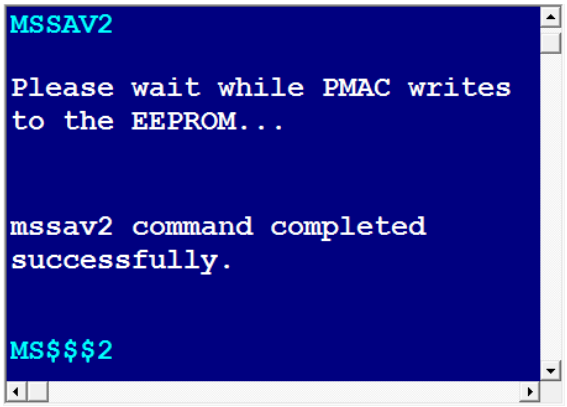
```
MS2,I11=1               ; Station number assignment (user configurable) for future
                        ; MACRO ASCII communication (e.g. MACSTA1)

MS2,I992=6527           ; Must be equal to the value of the ring controller's I6800
MS2,I997=0              ; Must be equal to the value of the ring controller's I6801

MS2,I995=$4080          ; Typical setting for MACRO slave device
MS2,I996=$0F8004        ; Nodes enabling, e.g. I/O node #2

MS2,I8=181              ; Ring check period (see equation below)
MS2,I9=28               ; Maximum ring error count  (see equation below)
MS2,I10=153             ; Minimum synch packet count  (see equation below)
```

These settings must be saved (e.g. MSSave2, or **MSSAV2**) on the ACC-65M, and followed by a reset (e.g. **MS$$$2**) to take effect:



If the clock settings are not at default, MS{},I8, I9, and I10 can be calculated using the following equations. Assuming a typical ring check period (RingCheckPeriod) of 20 milliseconds and a fatal packet error (MaxErrorPercent) of 15 percent:

$$\text{MS2, I8} = \text{INT}\left(\frac{(\text{Ringcheck Period} \times 117964.8) \times (\text{MS2, I997} + 1)}{(2 \times \text{MS2, I992} + 3)} + 1\right)$$

$$\text{MS2, I9} = \text{INT}\left(\frac{(\text{MS2, I8} \times \text{MaxErrorPe rcent})}{100} + 1\right)$$

$$\text{MS2, I10} = \text{MS2, I8} - \text{MS2, I9}$$

These must be computed explicitly ahead of time, expressions cannot be written directly into MS{} variables.

Step 4: I/O Data Registers

A single I/O node is sufficient for transferring the data to/from the ACC-65M. This is handled automatically in the firmware. The user's responsibility is choosing an available I/O node, enabling it per the example above, and finding the corresponding register (listed in the table below) for picking up the data.

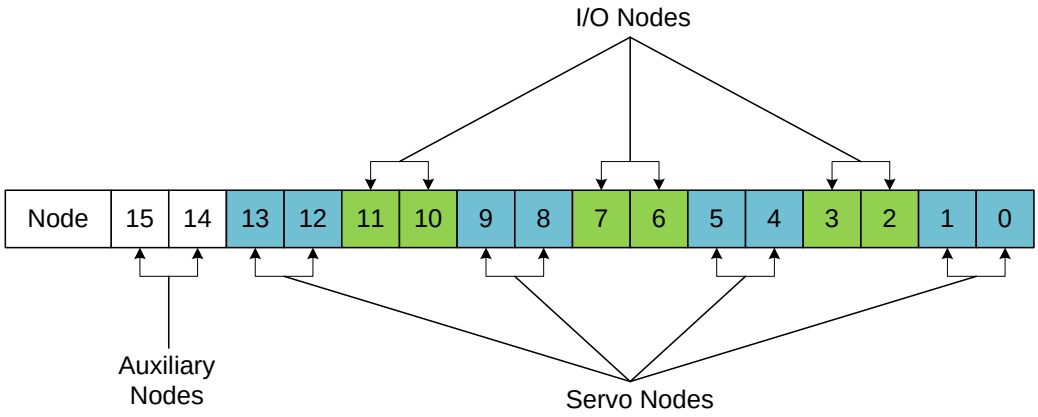
Nodes and Addressing

A Turbo PMAC, as a MACRO ring controller, can be populated with up to 4 MACRO ICs. This is reported by parameter **I4902**:

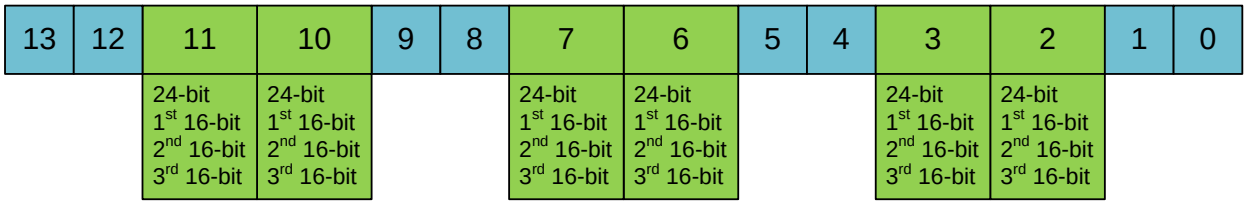
- = $0 No MACRO ICs (cannot be a ring controller)
- = $1 MACRO IC 0
- = $3 MACRO ICs 0 and 1
- = $7 MACRO ICs 0, 1, and 2
- = $F MACRO ICs 0, 1, 2, and 3

Each MACRO IC consists of 16 nodes: 2 auxiliary, 8 servo, and 6 I/O nodes.

- Auxiliary nodes are Master/Control registers and internal firmware use.
- Servo nodes carry information such as feedback, commands, and flags for motor control.
- I/O nodes are by default unoccupied and are user configurable for transferring miscellaneous data.



Each I/O node consists of 4 registers; one 24-bit and three 16-bit registers for a total of 72 bits of data.



Turbo Ring Controller I/O Node Registers

With the ACC-65M, we are only interested in the I/O data registers. The following, is a table of all the I/O node addresses of the ring controller for each MACRO IC:

	Ring Controller MACRO IC #0 Node Registers					
ACC-65M I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ring Controller I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
24-bit	X:$78420	X:$78424	X:$78428	X:$7842C	X:$78430	X:$78434
16-bit	X:$78421	X:$78425	X:$78429	X:$7842D	X:$78431	X:$78435
16-bit	X:$78422	X:$78426	X:$7842A	X:$7842E	X:$78432	X:$78436
16-bit	X:$78423	X:$78427	X:$7842B	X:$7842F	X:$78433	X:$78437

	Ring Controller MACRO IC #1 Node Registers					
ACC-65M I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ring Controller I/O Node#	**18**	**19**	**22**	**23**	**26**	**27**
24-bit	X:$79420	X:$79424	X:$79428	X:$7942C	X:$79430	X:$79434
16-bit	X:$79421	X:$79425	X:$79429	X:$7942D	X:$79431	X:$79435
16-bit	X:$79422	X:$79426	X:$7942A	X:$7942E	X:$79432	X:$79436
16-bit	X:$79423	X:$79427	X:$7942B	X:$7942F	X:$79433	X:$79437

	Ring Controller MACRO IC #2 Node Registers					
ACC-65M I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ring Controller I/O Node#	**34**	**35**	**38**	**39**	**42**	**43**
24-bit	X:$7A420	X:$7A424	X:$7A428	X:$7A42C	X:$7A430	X:$7A434
16-bit	X:$7A421	X:$7A425	X:$7A429	X:$7A42D	X:$7A431	X:$7A435
16-bit	X:$7A422	X:$7A426	X:$7A42A	X:$7A42E	X:$7A432	X:$7A436
16-bit	X:$7A423	X:$7A427	X:$7A42B	X:$7A42F	X:$7A433	X:$7A437

	Ring Controller MACRO IC #3 Node Registers					
ACC-65M I/O Node#	**2**	**3**	**6**	**7**	**10**	**11**
Ring Controller I/O Node#	**50**	**51**	**54**	**55**	**58**	**59**
24-bit	X:$7B420	X:$7B424	X:$7B428	X:$7B42C	X:$7B430	X:$7B434
16-bit	X:$7B421	X:$7B425	X:$7B429	X:$7B42D	X:$7B431	X:$7B435
16-bit	X:$7B422	X:$7B426	X:$7B42A	X:$7B42E	X:$7B432	X:$7B436
16-bit	X:$7B423	X:$7B427	X:$7B42B	X:$7B42F	X:$7B433	X:$7B437

Step 5: Using the ACC-65M Data

Having configured the following:

- Set up the MACRO ring controller
- Set up the phase clock to be the same across the ring
- Enabled a selected I/O node on the ACC-65M
- Enabled the corresponding I/O node on the ring controller side
- Saved and reset both the ACC-65M and the ring controller

The ACC-65M data should now be available to access from the ring controller side.

Digital Inputs and Outputs

The ACC-65M firmware transfers automatically the digitals inputs and outputs into/from the 24-bit register of the chosen I/O node. This is a read/write register, thus it is the same for both inputs and outputs.

I/O Node	Suggested M-Variable
2	M6977->X:$078420,0,24
3	M6978->X:$078424,0,24
6	M6979->X:$078428,0,24
7	M6980->X:$07842C,0,24
10	M6981->X:$078430,0,24
11	M6982->X:$078434,0,24
18	M6983->X:$079420,0,24
19	M6984->X:$079424,0,24
22	M6985->X:$079428,0,24
23	M6986->X:$07942C,0,24
26	M6987->X:$079430,0,24
27	M6988->X:$079434,0,24

I/O node	Suggested M-Variable
34	M6989->X:$078420,0,24
35	M6990->X:$07A424,0,24
38	M6991->X:$07A428,0,24
39	M6992->X:$07A42C,0,24
42	M6993->X:$07A430,0,24
43	M6994->X:$07A434,0,24
50	M6995->X:$07B420,0,24
51	M6996->X:$07B424,0,24
54	M6997->X:$07B428,0,24
55	M6998->X:$07B42C,0,24
58	M6999->X:$07B430,0,24
59	M7000->X:$07B434,0,24

Note

The inputs and outputs data registers are the same. These are read/write registers.

Inputs

The inputs can be simply mapped into the corresponding 24-bit register and queried at will. Direct bitwise mapping is possible for single I/O point access. For example, using I/O node #2:

ACC-65M Digital Inputs (node #2)

```
#define Input1 M7001    Input1->X:$078420,0,1
#define Input2 M7002    Input2->X:$078420,1,1
#define Input3 M7003    Input3->X:$078420,2,1
#define Input4 M7004    Input4->X:$078420,3,1
#define Input5 M7005    Input5->X:$078420,4,1
#define Input6 M7006    Input6->X:$078420,5,1
#define Input7 M7007    Input7->X:$078420,6,1
#define Input8 M7008    Input8->X:$078420,7,1
#define Input9 M7009    Input9->X:$078420,8,1
#define Input10 M7010   Input10->X:$078420,9,1
#define Input11 M7011   Input11->X:$078420,10,1
#define Input12 M7012   Input12->X:$078420,11,1
#define Input13 M7013   Input13->X:$078420,12,1
#define Input14 M7014   Input14->X:$078420,13,1
#define Input15 M7015   Input15->X:$078420,14,1
#define Input16 M7016   Input16->X:$078420,15,1
#define Input17 M7017   Input17->X:$078420,16,1
#define Input18 M7018   Input18->X:$078420,17,1
#define Input19 M7019   Input19->X:$078420,18,1
#define Input20 M7020   Input20->X:$078420,19,1
#define Input21 M7021   Input21->X:$078420,20,1
#define Input22 M7022   Input22->X:$078420,21,1
#define Input23 M7023   Input23->X:$078420,22,1
#define Input24 M7024   Input24->X:$078420,23,1
```

Outputs

The outputs require:

- Writing to the whole 24-bit register
- An image word for reporting the status to the user

This will be done in a simple PLC logic.

Note

It is possible to write to the I/O node register bits individually, but not more than one at the time. Thus, the use of an image word.

For creating an image word, it is suggested to use one of the open memory registers in Turbo ($10F0 - $10FF) which can be either X or Y. For example, using X:$10FF with I/O node 2:

ACC-65M Digital Outputs Mapping

#define Output1 M7025	Output1->X:$10FF,0,1
#define Output2 M7026	Output2->X:$10FF,1,1
#define Output3 M7027	Output3->X:$10FF,2,1
#define Output4 M7028	Output4->X:$10FF,3,1
#define Output5 M7029	Output5->X:$10FF,4,1
#define Output6 M7030	Output6->X:$10FF,5,1
#define Output7 M7031	Output7->X:$10FF,6,1
#define Output8 M7032	Output8->X:$10FF,7,1
#define Output9 M7033	Output9->X:$10FF,8,1
#define Output10 M7034	Output10->X:$10FF,9,1
#define Output11 M7035	Output11->X:$10FF,10,1
#define Output12 M7036	Output12->X:$10FF,11,1
#define Output13 M7037	Output13->X:$10FF,12,1
#define Output14 M7038	Output14->X:$10FF,13,1
#define Output15 M7039	Output15->X:$10FF,14,1
#define Output16 M7040	Output16->X:$10FF,15,1
#define Output17 M7041	Output17->X:$10FF,16,1
#define Output18 M7042	Output18->X:$10FF,17,1
#define Output19 M7043	Output19->X:$10FF,18,1
#define Output20 M7044	Output20->X:$10FF,19,1
#define Output21 M7045	Output21->X:$10FF,20,1
#define Output22 M7046	Output22->X:$10FF,21,1
#define Output23 M7047	Output23->X:$10FF,22,1
#define Output24 M7048	Output24->X:$10FF,23,1

The following PLC will then copy the outputs (from the open memory register) into the 24-bit I/O node register. A simple latch is used to prevent the PLC from overwhelming the I/O register:

```
#define N2Twenty4      M6977   ; Node 2, 24-bit data register
#define OutMirror      M7049   ; Outputs Mirror register

N2Twenty4->X:$078420,0,24      ; Node 2, 24-bit data register
OutMirror->X:$10FF,0,24        ; Open memory register (user configurable)
OutMirror = 0                  ; Initialize/save to zero or desired state

Open plc 1 clear
N2Twenty4 = OutMirror          ; Update data register
Close
```

With this PLC executing constantly, the user now writes to M4051 through M4074 to toggle the digital outputs on the ACC-65M.

Analog Inputs (ADCS)

The ACC-65M firmware transfers automatically the analog inputs into/from the 1st and 2nd 16-bit registers of the chosen I/O node. These are read/write registers:

Newer revision of the ACC-65M (3-pin edge logic connector)					
I/O	**ADC #1**	**ADC #2**	**I/O**	**ADC #1**	**ADC #2**
2	M5001->X:$078421,8,16,S	M5002->X:$078422,8,16,S	**34**	M5025->X:$07A421,8,16,S	M5026->X:$07A422,8,16,S
3	M5003->X:$078425,8,16,S	M5004->X:$078426,8,16,S	**35**	M5027->X:$07A425,8,16,S	M5028->X:$07A426,8,16,S
6	M5005->X:$078429,8,16,S	M5006->X:$07842A,8,16,S	**38**	M5029->X:$07A429,8,16,S	M5030->X:$07A42A,8,16,S
7	M5007->X:$07842D,8,16,S	M5008->X:$07842E,8,16,S	**39**	M5031->X:$07A42D,8,16,S	M5032->X:$07A42E,8,16,S
10	M5009->X:$078431,8,16,S	M5010->X:$078432,8,16,S	**42**	M5033->X:$07A431,8,16,S	M5034->X:$07A432,8,16,S
11	M5011->X:$078435,8,16,S	M5012->X:$078436,8,16,S	**43**	M5035->X:$07A435,8,16,S	M5036->X:$07A436,8,16,S
18	M5013->X:$079421,8,16,S	M5014->X:$079422,8,16,S	**50**	M5037->X:$07B421,8,16,S	M5038->X:$07B422,8,16,S
19	M5015->X:$079425,8,16,S	M5016->X:$079426,8,16,S	**51**	M5039->X:$07B425,8,16,S	M5040->X:$07B426,8,16,S
22	M5017->X:$079429,8,16,S	M5018->X:$07942A,8,16,S	**54**	M5041->X:$07B429,8,16,S	M5042->X:$07B42A,8,16,S
23	M5019->X:$07942D,8,16,S	M5020->X:$07942E,8,16,S	**55**	M5043->X:$07B42D,8,16,S	M5044->X:$07B42E,8,16,S
26	M5021->X:$079431,8,16,S	M5022->X:$079432,8,16,S	**58**	M5045->X:$07B431,8,16,S	M5046->X:$07B432,8,16,S
27	M5023->X:$079435,8,16,S	M5024->X:$079436,8,16,S	**59**	M5047->X:$07B435,8,16,S	M5048->X:$07B436,8,16,S

Note

The ADCs on the older revision of the ACC-65M (2-pin Molex logic connector) are 12 bits.

Older revision of the ACC-65M (2-pin Molex logic connector)					
I/O	**ADC #1**	**ADC #2**	**I/O**	**ADC #1**	**ADC #2**
2	M5001->X:$078421,12,12,S	M5002->X:$078422,12,12,S	**34**	M5025->X:$07A421,12,12,S	M5026->X:$07A422,12,12,S
3	M5003->X:$078425,12,12,S	M5004->X:$078426,12,12,S	**35**	M5027->X:$07A425,12,12,S	M5028->X:$07A426,12,12,S
6	M5005->X:$078429,12,12,S	M5006->X:$07842A,12,12,S	**38**	M5029->X:$07A429,12,12,S	M5030->X:$07A42A,12,12,S
7	M5007->X:$07842D,12,12,S	M5008->X:$07842E,12,12,S	**39**	M5031->X:$07A42D,12,12,S	M5032->X:$07A42E,12,12,S
10	M5009->X:$078431,12,12,S	M5010->X:$078432,12,12,S	**42**	M5033->X:$07A431,12,12,S	M5034->X:$07A432,12,12,S
11	M5011->X:$078435,12,12,S	M5012->X:$078436,12,12,S	**43**	M5035->X:$07A435,12,12,S	M5036->X:$07A436,12,12,S
18	M5013->X:$079421,12,12,S	M5014->X:$079422,12,12,S	**50**	M5037->X:$07B421,12,12,S	M5038->X:$07B422,12,12,S
19	M5015->X:$079425,12,12,S	M5016->X:$079426,12,12,S	**51**	M5039->X:$07B425,12,12,S	M5040->X:$07B426,12,12,S
22	M5017->X:$079429,12,12,S	M5018->X:$07942A,12,12,S	**54**	M5041->X:$07B429,12,12,S	M5042->X:$07B42A,12,12,S
23	M5019->X:$07942D,12,12,S	M5020->X:$07942E,12,12,S	**55**	M5043->X:$07B42D,12,12,S	M5044->X:$07B42E,12,12,S
26	M5021->X:$079431,12,12,S	M5022->X:$079432,12,12,S	**58**	M5045->X:$07B431,12,12,S	M5046->X:$07B432,12,12,S
27	M5023->X:$079435,12,12,S	M5024->X:$079436,12,12,S	**59**	M5047->X:$07B435,12,12,S	M5048->X:$07B436,12,12,S

Testing the Analog Inputs

The ADC input data is typically signed (bipolar). If the ACC-65M is set, by jumpers, to read unsigned (unipolar) data then the M-Variable definitions should be changed to Mxxx->X:$xxxxxx,16,16,**U**.

With the 16-bit ADCs, the user should expect to see:

		Single-Ended Signal [VDC]	**Differential Signal [VDC]**	**Software Counts**
Unipolar	**Bipolar**	-10	-5	-32768
		0	0	0
		10	5	32768

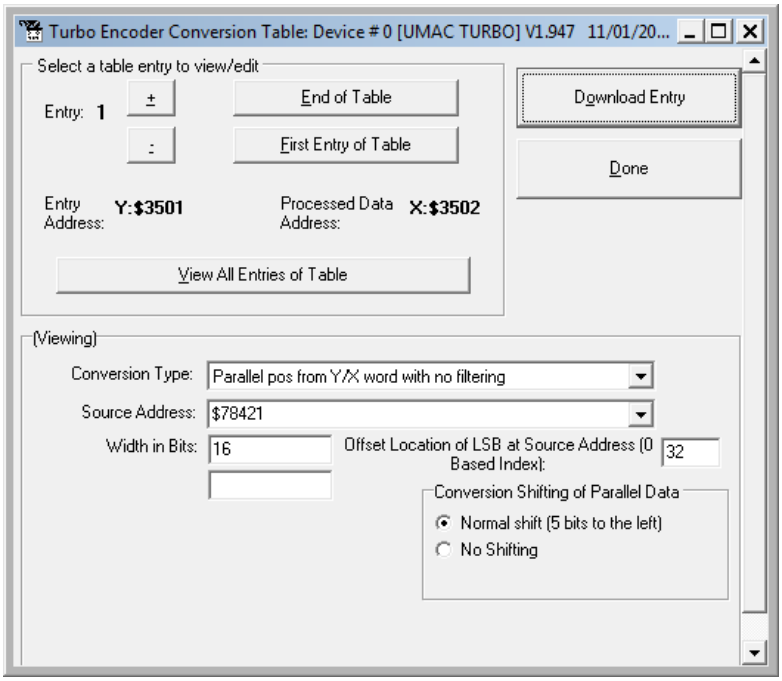
With the 12-bit ADCs, the user should expect to see:

		Single-Ended Signal [VDC]	**Differential Signal [VDC]**	**Software Counts**
Unipolar	**Bipolar**	-10	-5	-2048
		0	0	0
		10	5	2048

Using the ADCs for Servo Feedback

Using an analog ADC input for servo requires bringing it into the encoder conversion table (ECT). Using the automatic ECT utility in the PeWin32Pro2 software:

- Type: Parallel position from Y/X
- Source Address: I/O node data register (i.e. $78421)
- Width in bits: 16 (16-bit ADC)
- Offset of LSB: 32 (because it is an X register)
- Normal shift


Turbo Encoder Conversion Table: Device # 0 [UMAC TURBO] V1.947 11/01/20...
Select a table entry to view/edit
Entry: 1
+
-
End of Table
First Entry of Table
Download Entry
Done
Entry Address: Y:$3501
Processed Data Address: X:$3502
View All Entries of Table
(Viewing)
Conversion Type: Parallel pos from Y/X word with no filtering
Source Address: $78421
Width in Bits: 16
Offset Location of LSB at Source Address (0 Based Index): 32
Conversion Shifting of Parallel Data
Normal shift (5 bits to the left)
No Shifting

Alternately, using the equivalent I8000 parameters:

```
I8000 = $678421        ; @$3501
I8001 = $10020         ; @$3502
```

The ADC data is now processed in the encoder conversion table.

A motor can use it for position/velocity feedback
Example: Ixx03=$3502, Ixx04=$3502

Or as a master position
Example: Ixx05=$3502

Otherwise it can be accessed manually using an M-Variable pointer. Note that you would need to divide by 32 or 2^5 for proper scaling:
Example: M1000->X;$3502,0,24,S

Analog Outputs (DACs)

The analog outputs (DACs) map out to the same registers as the analog inputs (ADCs). These are read write registers. The ACC-65M firmware handles automatically the transfer of this data.

ACC-65M Analog Output Data Registers					
I/O	**DAC #1**	**DAC #2**	**I/O**	**DAC #1**	**DAC #2**
2	M6001->X:$078421,8,16,S	M6002->X:$078422,8,16,S	**34**	M6025->X:$07A421,8,16,S	M6026->X:$07A422,8,16,S
3	M6003->X:$078425,8,16,S	M6004->X:$078426,8,16,S	**35**	M6027->X:$07A425,8,16,S	M6028->X:$07A426,8,16,S
6	M6005->X:$078429,8,16,S	M6006->X:$07842A,8,16,S	**38**	M6029->X:$07A429,8,16,S	M6030->X:$07A42A,8,16,S
7	M6007->X:$07842D,8,16,S	M6008->X:$07842E,8,16,S	**39**	M6031->X:$07A42D,8,16,S	M6032->X:$07A42E,8,16,S
10	M6009->X:$078431,8,16,S	M6010->X:$078432,8,16,S	**42**	M6033->X:$07A431,8,16,S	M6034->X:$07A432,8,16,S
11	M6011->X:$078435,8,16,S	M6012->X:$078436,8,16,S	**43**	M6035->X:$07A435,8,16,S	M6036->X:$07A436,8,16,S
18	M6013->X:$079421,8,16,S	M6014->X:$079422,8,16,S	**50**	M6037->X:$07B421,8,16,S	M6038->X:$07B422,8,16,S
19	M6015->X:$079425,8,16,S	M6016->X:$079426,8,16,S	**51**	M6039->X:$07B425,8,16,S	M6040->X:$07B426,8,16,S
22	M6017->X:$079429,8,16,S	M6018->X:$07942A,8,16,S	**54**	M6041->X:$07B429,8,16,S	M6042->X:$07B42A,8,16,S
23	M6019->X:$07942D,8,16,S	M6020->X:$07942E,8,16,S	**55**	M6043->X:$07B42D,8,16,S	M6044->X:$07B42E,8,16,S
26	M6021->X:$079431,8,16,S	M6022->X:$079432,8,16,S	**58**	M6045->X:$07B431,8,16,S	M6046->X:$07B432,8,16,S
27	M6023->X:$079435,8,16,S	M6024->X:$079436,8,16,S	**59**	M6047->X:$07B435,8,16,S	M6048->X:$07B436,8,16,S

Note

Although the DACs are 12-bit filtered PWM they are mapped to the upper 16 bits.

Image Word

Since these are read write registers, and in order for the user to report the value of the analog output(s) in software, a simple image word PLC must be written.

For creating an image word, it is suggested to use one of the open memory registers in Turbo $10F0 - $10FF which can be either X or Y. For example, using Y:$10FE and X:$10FE to mirror image DAC 1 and DAC 2 respectively at Node 2:

```
#define N2First16      M6001
#define N2Second16     M6002
N2First16->X:$078421,8,16,S   ; Node 2, 1st 16-bit register
N2Second16->X:$078422,8,16,S  ; Node 2, 2nd 16-bit register

#define DAC1   M5091          ; Node 2 DAC 1
#define DAC2   M5092          ; Node 2 DAC 2
DAC1->Y:$10FF,8,16,S          ; Image word using open memory
DAC2->X:$10FE,8,16,S          ; Image word using open memory
DAC1 = 0                      ; Save/Initialize to zero or desired state
DAC2 = 0                      ; Save/Initialize to zero or desired state

Open plc 1 clear
N2First16  = DAC1      ; Update data register, DAC 1
N2Second16 = DAC2      ; Update data register, DAC 2
Close
```

With this PLC executing constantly, the user now writes to DAC1 and DAC2 M-Variables to manipulate the voltage outputs of the analog outputs.

Testing the Analog Outputs

These are ±10V outputs, where 10 volts corresponds to the value of **MS2,I992**. Remember that this is dictated by the ring phase clock, do not attempt to change it in this section.

With the default clock setting (e.g. MS2,I992=6527), and by writing to the analog output data register (e.g. using the suggested M-Variable), the user should see:

Suggested M-Variable	Single Ended [VDC]	Differential [VDC]
-6527	-10	-20
-3264	-5	-10
0	0	0
3264	+5	+10
6527	+10	+20

General Purpose Relay Outputs

The relay outputs are mapped into bits 19, and 20 of the 3rd 16-bit register of the I/O node.

ACC-65M GP Relays Data Registers					
I/O	**GP Relay 1**	**GP Relay 2**	**I/O**	**GP Relay 1**	**GP Relay 2**
2	M6051->X:$078423,19	M6052->X:$078423,20	**34**	M6075->X:$07A423,19	M6076->X:$07A423,20
3	M6053->X:$078427,19	M6054->X:$078427,20	**35**	M6077->X:$07A427,19	M6078->X:$07A427,20
6	M6055->X:$07842B,19	M6056->X:$07842B,20	**38**	M6079->X:$07A42B,19	M6080->X:$07A42B,20
7	M6057->X:$07842F,19	M6058->X:$07842F,20	**39**	M6081->X:$07A42F,19	M6082->X:$07A42F,20
10	M6059->X:$078433,19	M6060->X:$078433,20	**42**	M6083->X:$07A433,19	M6084->X:$07A433,20
11	M6061->X:$078437,19	M6062->X:$078437,20	**43**	M6085->X:$07A437,19	M6086->X:$07A437,20
18	M6063->X:$079423,19	M6064->X:$079423,20	**50**	M6087->X:$07B423,19	M6088->X:$07B423,20
19	M6065->X:$079427,19	M6066->X:$079427,20	**51**	M6089->X:$07B427,19	M6090->X:$07B427,20
22	M6067->X:$07942B,19	M6068->X:$07942B,20	**54**	M6091->X:$07B42B,19	M6092->X:$07B42B,20
23	M6069->X:$07942F,19	M6070->X:$07942F,20	**55**	M6093->X:$07B42F,19	M6094->X:$07B42F,20
26	M6071->X:$079433,19	M6072->X:$079433,20	**58**	M6095->X:$07B433,19	M6096->X:$07B433,20
27	M6073->X:$079437,19	M6074->X:$079437,20	**59**	M6097->X:$07B437,19	M6098->X:$07B437,20

It is possible to toggle these relay outputs separately or simultaneously in software by writing to the full 16-bit word. However, due to the read/write nature of the I/O node register they cannot be written to simultaneously using two independent bits (e.g. terminal command: M6051=1 M6052=1). Two mirror image bits can be used to make this possible.

For creating an image word, it is suggested to use one of the open memory registers in Turbo $10F0 - $10FF which can be either X or Y. For example, using bits 19, and 20 of Y:$10FF to mirror image Relay 1 and Relay 2 at Node 2:

```
#define N2Third16Bit19 M6099
#define N2Third16Bit20 M6100
N2Third16Bit19->X:$078423,19  ; Node 2, third 16-bit register, bit 19
N2Third16Bit20->X:$078423,20  ; Node 2, third 16-bit register, bit 20

#define Relay1         M6101
#define Relay2         M6102
Relay1->Y:$10FF,19            ; Mirror Bit Relay 1
Relay2->Y:$10FF,20            ; Mirror Bit Relay 2
Relay1 = 0                    ; Save/Initialize to zero or desired state
Relay2 = 0                    ; Save/Initialize to zero or desired state

Open PLC 1 Clear
N2Third16Bit19 = Relay1
N2Third16Bit20 = Relay2
Close
```

Testing the General Purpose Relays

The following table summarizes the relay functions. That is the relationship between the common line and the normally open / normally closed lines:

GP Relay 1	Connection between pins #13 (COM) and #14 (NO)	Connection between pins #13 (COM) and #6 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

GP Relay 2	Connection between pins #7 (COM) and #8 (NO)	Connection between pins #7 (COM) and #15 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

CONNECTOR PINOUTS AND WIRING

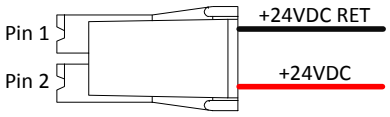
24 VDC Input

This connector is used to bring in the 24 VDC logic power supply. It must be able to provide an instantaneous current of about 2 amperes.

With the new revision of the ACC-65M, using the Phoenix PCB edge connector, the power supply for the digital and analog outputs is brought in separately through pin #3. This allows the logic power to stay on in the case of a fuse trip (too much current draw out of the outputs).

This connection can be made using a 16 AWG wire directly from a protected power supply. In situations where the power supply is shared with other devices, it may be desirable to insert a filter in this connection. Also, it is highly recommended that each device be wired back to the power supply terminals independently.

Phoenix PCB Edge Connector 3-pin connector				
Pin #	**Symbol**	**Function**	**Description**	**Notes**
1	+24 VDC RET	Common	Logic power return	
2	+24 VDC Control	Input	Logic power supply	24 VDC +/-10%, 2A
3	+24 VDC PWR	Input	Digital/Analog outputs power supply	12 - 24 VDC
Phoenix part #: ZEC 1,5/ 3-ST-5,0 C2 R1,3 (18883051) Delta Tau part #: 014-188305-001 (For Internal Use)				

ACC-65M revision 101 and older		
Connector: Molex 2-pin Female Mating: Molex 2-pin Male		
Pin #	**Symbol**	**Function**
1	+24 VDC RET	Logic power return
2	+24 VDC PWR	+24VDC logic and outputs power supply

Molex Mating Connector p/n:	0444412002
Molex Crimper tool p/n:	63811-0400
Molex Pins p/n:	0433751001
Delta Tau Mating Connector p/n:	014-000F02-HSG
Delta Tau Pins p/n:	014-043375-001

Digital Inputs

Connector: Phoenix Contact MSTB 2.5-5.08, 15 Positions (female) **Mating: Phoenix Contact LR13631 (male)**		
Pin #	**Symbol**	**Function**
1	IN01	INPUT 1
2	IN02	INPUT 2
3	IN03	INPUT 3
4	IN04	INPUT 4
5	RET	RETURN FOR INPUTS 1-4
6	IN05	INPUT 5
7	IN06	INPUT 6
8	IN07	INPUT 7
9	IN08	INPUT 8
10	RET	RETURN FOR INPUTS 5-8
11	IN09	INPUT 9
12	IN10	INPUT 10
13	IN11	INPUT 11
14	IN12	INPUT 12
15	RET	RETURN FOR INPUTS 9-12
16	IN13	INPUT 13
17	IN14	INPUT 14
18	IN15	INPUT 15
19	IN16	INPUT 16
20	RET	RETURN FOR INPUTS 13-16
21	IN17	INPUT 17
22	IN18	INPUT 18
23	IN19	INPUT 19
24	IN20	INPUT 20
25	RET	RETURN FOR INPUTS 17-20
26	IN21	INPUT 21
27	IN22	INPUT 22
28	IN23	INPUT 23
29	IN24	INPUT 24
30	RET	RETURN FOR INPUTS 21-24

Note

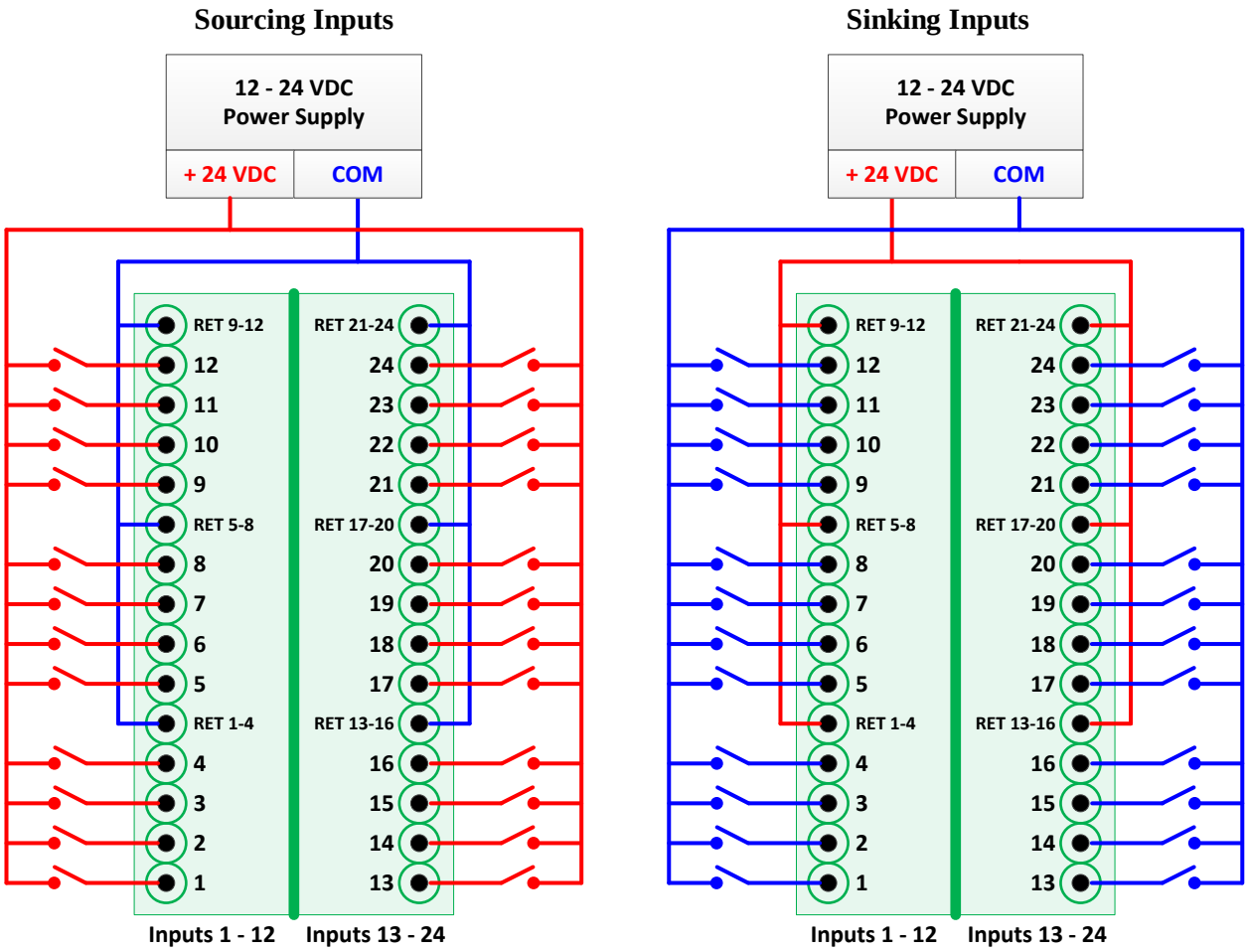
All the inputs return lines are internally tied together. They are labeled for each set of four inputs for wiring convenience.

Wiring the digital Inputs

The inputs can be wired to be either sinking into or sourcing out of the ACC-65M.

For sourcing, connect the +24V side of the power supply to the individual input switches and the GND side to the corresponding return line.

For sinking, connect the GND side of the power supply to the individual input switches and the +24V side to the corresponding return line.



Digital Outputs

Connector: Phoenix Contact MSTB 2.5-5.08, 15 Positions (female) **Mating: Phoenix Contact LR13631 (male)**		
Pin #	**Symbol**	**Function**
1	OUT 01	OUTPUT 1
2	OUT 02	OUTPUT 2
3	OUT 03	OUTPUT 3
4	OUT 04	OUTPUT 4
5	RET	RETURN FOR OUTPUTS 1-4
6	OUT 05	OUTPUT 5
7	OUT 06	OUTPUT 6
8	OUT 07	OUTPUT 7
9	OUT 08	OUTPUT 8
10	RET	RETURN FOR OUTPUTS 5-8
11	OUT 09	OUTPUT 9
12	OUT 10	OUTPUT 10
13	OUT 11	OUTPUT 11
14	OUT 12	OUTPUT 12
15	RET	RETURN FOR OUTPUTS 9-12
16	OUT 13	OUTPUT 13
17	OUT 14	OUTPUT 14
18	OUT 15	OUTPUT 15
19	OUT 16	OUTPUT 16
20	RET	RETURN FOR OUTPUTS 13-16
21	OUT 17	OUTPUT 17
22	OUT 18	OUTPUT 18
23	OUT 19	OUTPUT 19
24	OUT 20	OUTPUT 20
25	RET	RETURN FOR OUTPUTS 17-20
26	OUT 21	OUTPUT 21
27	OUT 22	OUTPUT 22
28	OUT 23	OUTPUT 23
29	OUT 24	OUTPUT 24
30	RET	RETURN FOR OUTPUTS 21-24

Note

All the outputs return lines are internally tied together. They are labeled for each set of four outputs for wiring convenience.

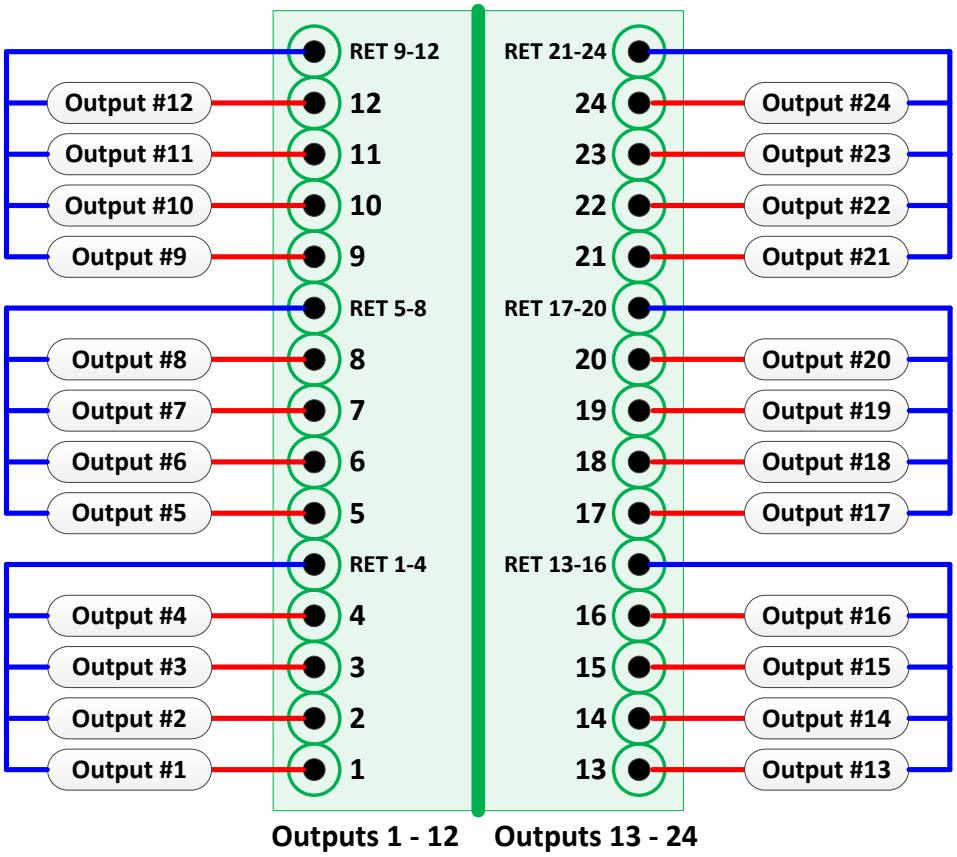
Wiring the digital outputs

The outputs are always sourcing in the ACC-65M.

Caution

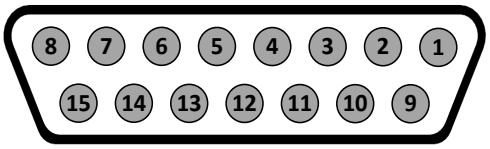
The maximum current draw out of each output load is not to exceed 600 mA @ 24VDC.

The return lines (pins #5, 10, 15, 20, 25 and 30) are all internally connected. The diagram shows them in sets of four for wiring convenience.



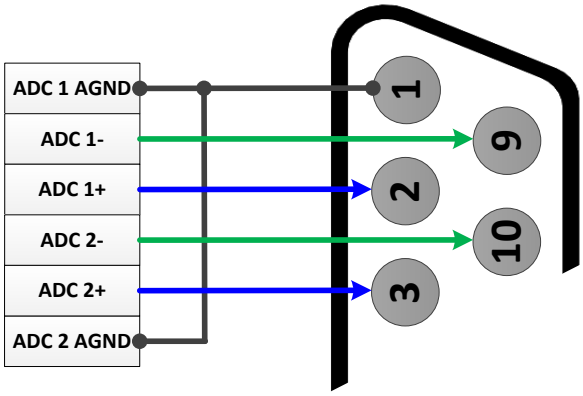
Analog Connector

This optional connector provides connections to the analog outputs and inputs, as well as the general purpose relays.

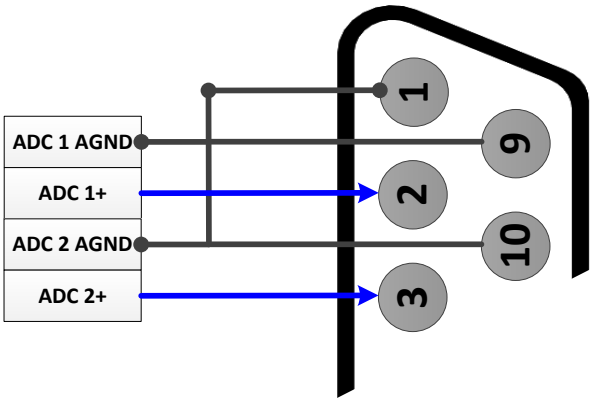
Connector: D-sub DA-15F **Mating: D-sub DA-15M**		
Pin #	**Symbol**	**Function**
1	AGND	Common Analog Ground
2	ADC 1+	Analog Input 1+
3	ADC 2+	Analog Input 2+
4	DAC 1+	Analog Output 1+
5	DAC 2+	Analog Output 2+
6	AE-NC 1	Normally closed Relay 1
7	AE-COM 2	Relay 2 Common
8	AE-NO 2	Normally open Relay 2
9	ADC 1-	Analog Input 1-
10	ADC 2-	Analog Input 2-
11	DAC 1-	Analog Output 1-
12	DAC 2-	Analog Output 2-
13	AE-COM 1	Relay 1 Common
14	AE-NO 1	Normally open Relay 1
15	AE-NC 2	Normally closed Relay 2

Wiring the Analog (ADC) Inputs

Differential Analog Input Signals



Single Ended Analog Input Signals



Note

For single-ended connections, tie the negative ADC pin to ground.

Note

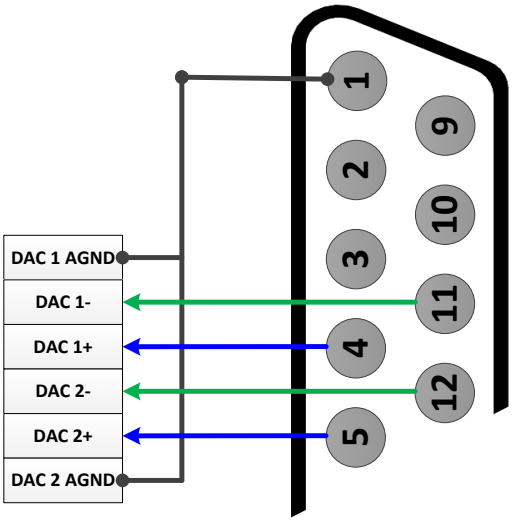
The analog inputs use the **ADS8321** Converter device

Note

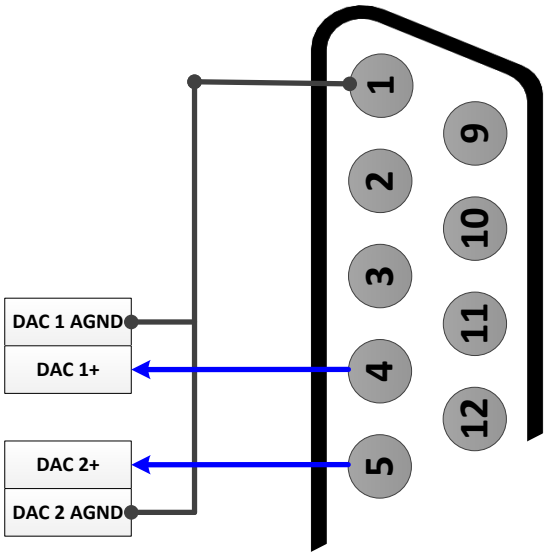
Full (16-bit) resolution is available for bipolar signals only. Half of the range of the full resolution is used for unipolar (0 - 5V or 0 - 10V) signals.

Wiring the Analog (DAC) Outputs

Differential DAC Output Signals



Single Ended DAC Output Signal



Wiring the General Purpose Relays

The general purpose relays provide a signal which can be either:

- High true using the normally open contact (pins #14, and 8 respectively)
- Low true using the normally closed contact (pin #6, and 15 respectively)

Also, they can be either sourcing or sinking depending on the wiring scheme (common line).

The following table summarizes the relay functions. That is the relationship between the common line and the normally open / normally closed lines:

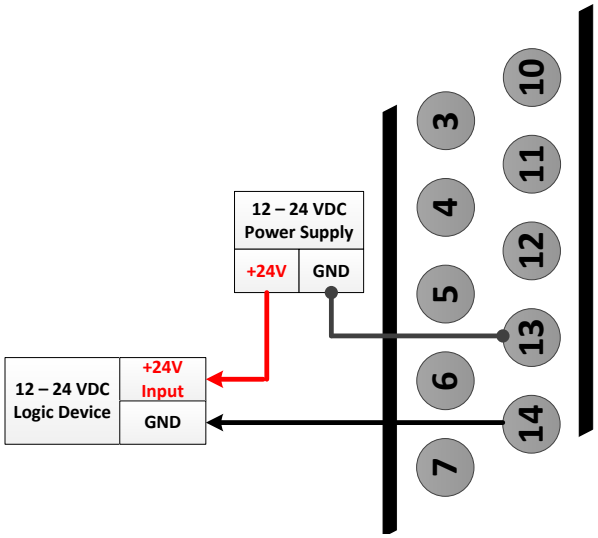
GP Relay 1	Connection between pins #13 (COM) and #14 (NO)	Connection between pins #13 (COM) and #6 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

GP Relay 2	Connection between pins #7 (COM) and #8 (NO)	Connection between pins #7 (COM) and #15 (NC)
Software bit = 0	Open	Closed
Software bit = 1	Closed	Open

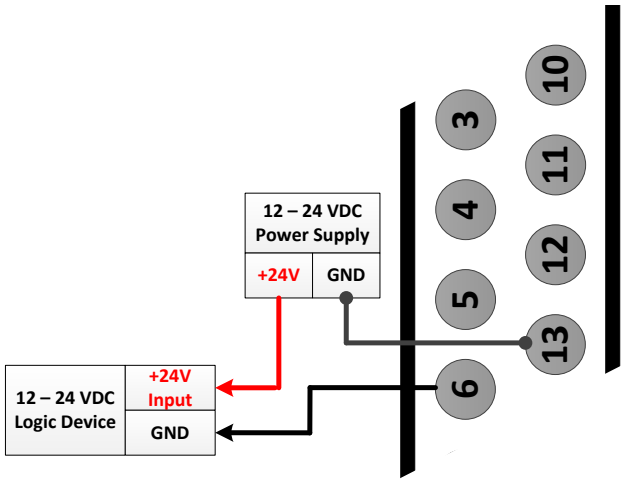
Below, are wiring samples using general purpose relay 1:

Sourcing

High True Output to Logic Device

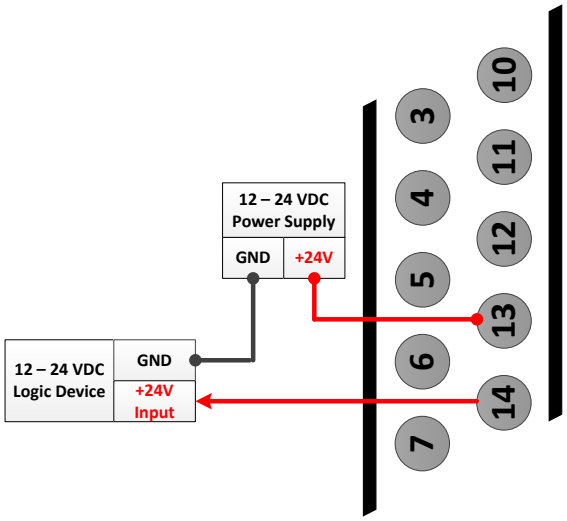


Low True Output to Logic Device

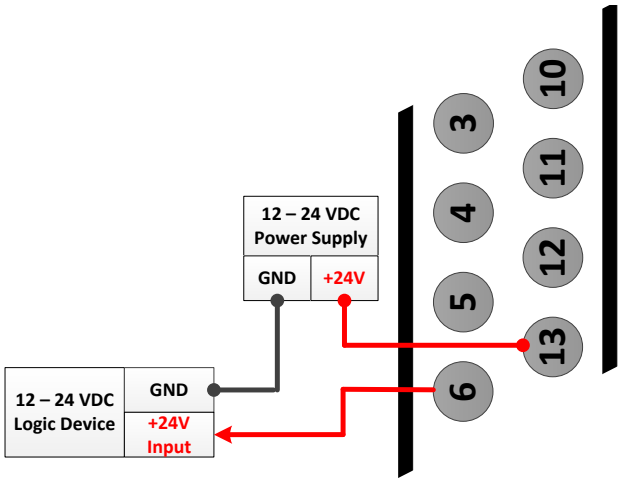


Sinking

High True Output to Logic Device

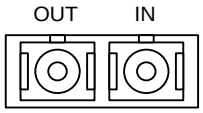


Low True Output to Logic Device

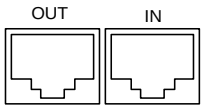


MACRO Connection

These connections are used to connect the MACRO cables to the ACC-65M.

Fiber Connector		
Pin #	**Symbol**	**Function**
1	IN	MACRO Ring Receiver
2	OUT	MACRO Ring Transmitter

The fiber optic cables, typically acquired from Delta Tau, are 62.5/125 multi-mode glass fiber terminated in an SC-style connector, with an optical wavelength of 1,300 nm.

RJ45 CAT5e			
Pin #	**Symbol**	**Function**	**Description**
1	DATA+	Data +	Differential MACRO Signal
2	DATA-	Data -	Differential MACRO Signal
3 – 8	Unused	-	Unused terminated pins

The RJ-45 cable used for MACRO is CAT5 verified straight-through 8-conductor.

The input (IN) connector of the ACC-65M is inserted into the MACRO output (OUT) connector of the previous device on the MACRO ring.

The output (OUT) connector of the ACC-65M is inserted into the input (IN) MACRO connector of the next device on the MACRO ring.

Universal Serial Bus (USB)

The USB port is used to change/reload the operational firmware of the ACC-65M. It utilizes a USB A-B type cable to make the connection between the ACC-65M and a host PC.

This connection appears in the hardware device manager of the PC under the serial communication port(s). Typically, the firmware is downloaded using Delta Tau's MACRO firmware utility.

Pin	Symbol	Function
1	VCC	N.C.
2	D-	DATA-
3	D+	DATA+
4	GND	GND
5	SHELL	SHIELD
6	SHELL	SHIELD

The serial port settings should be as follows:
Baud Rate: 9600 if jumper E3 is installed, 38400 if jumper E3 is not installed (default)
Data Bits: 8
Parity: None
Stop Bits: 1
Flow Control: Xon/Xoff

Note

The PeWin32PRO2 software must be installed for the PC to recognize the serial communication connection.

TROUBLESHOOTING

Initializing the ACC-65M, Clearing Faults

Typically, peripherals such as the ACC-65M are powered up first (before the ring controller) in a MACRO ring configuration. So that when the ring controller comes up, it clears any MACRO faults and initializes the ring. This would be equivalent to a ring controller reset ($$$).

If the ACC-65M is powered up at the same time or after the ring controller has been turned on, then it is advised to implement a reset mechanism in software (flag) or hardware (e.g. push button) to clear any MACRO ring faults and initialize the ring. Essentially, the ring controller must issue:

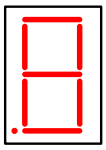
- MSCRLF{any slave enabled node}
- CLRF

The simplest implementation can be done in the startup PLC, which is enabled or released using a conditional flag when all the hardware is powered up and ready. For example:

```
P8000 = 0     ; Flag to clear MACRO ring faults
Open plc 1 Clear
If (P8000 = 1); Clear faults?
 I5111 = 250 * 8388608 / I10 While (I5111 >0) EndW ; 250 msec delay
 CMD"MSCLRF15"; Broadcast a "clear faults" on all enabled nodes of MACRO IC 0
 I5111 =  50 * 8388608 / I10 While (I5111 >0) EndW ; 50 msec delay
 CMD"CLRF"    ; Clear ring controller MACRO ring faults
 I5111 =  50 * 8388608 / I10 While (I5111 >0) EndW ; 50 msec delay
 P8000 = 0    ; Reset flag
EndIF
Close
```

Error Codes (7-Segment LED)

This 7-Segment LED Indicator reports the error and MACRO status of the ACC-65M.



Code	Fault	Notes
0	Ring Active	No errors. Normal operation mode
1 – 9	N/A	
A	24V Input	Check the 24V logic power input
B	Ring Break	Indicates a MACRO ring break: • MACRO cables on are unplugged or broken. • MACRO cables not connected in the correct order (In/Out)
C	Configuration	Indicates that the software settings do not match physical hardware: • Enabled node on the ACC-65M not enabled on the ring controller side • Check node settings on both the ACC-65M and ring controller
D	Data Error	Indicates a packet loss or other MACRO data loss • Verify the setting of I80, I81, and I82 on the ring controller • Verify the setting of MS{}, I8, I9 and I10 on the ACC-65M
E	N/A	
F	Ring Fault	Indicates that a momentary MACRO ring fault has occurred: • Verify the setting of I80, I81, and I82 on the ring controller

LED Status

Input and Output LED Indicators

Each of the 24 input and 24 output lines has an associated LED on the front panel of the unit that displays its current state: either active (in a Green or Red state) or inactive (darkened; no light).

Status LED

+24V: when lit, this LED indicates that the I/O 24V power is applied

Fuse: when lit, this LED indicates that the internal fuse protecting the external 24V is properly functional

PWR: when lit, this LED indicates that the 24V logic power is applied

WD: when lit, this LED indicates that the watchdog safety circuit is activated.
This indicates a failure condition and interrupts MACRO communication. Also, turns off all outputs (Digital and Analog). It occurs if any of the following is true:

- CPU over-clocked
  In this mode, the CPU indicates that is has been overloaded with computation and cannot accomplish tasks in a timely manner. The only possible culprit with the ACC-65M is the user programmable code (locally stored on the ACC-65M) PLCC.

- Incorrect clock settings
  The phase clock setting on the ACC-65M does not match the ring controller's phase clock.

- Hardware +5V failure (internal) or short
  In this mode, the internal 5V logic circuitry has failed. Check PWR Led Status.

Relay Status LED

RLY1: when lit, this LED indicates that the first amplifier enable relay is activated

RLY2: when lit, this LED indicates that the second amplifier enable relay is activated

MACRO Link LED

Green: Indicates that the MACRO ring is properly wired

Red: Indicates a ring break, or MACRO ring cables not connected in the correct order (IN/ OUT).

APPENDIX A: MEMORY MAP

PMAC3 Style ASIC

The Power PMAC CPU can interface with up to 16 PMAC3 Style ASICs which base addresses are:

Index	Card I/O Address	Index	Card I/O Address
Gate3[0]	$900000	Gate3[8]	$920000
Gate3[1]	$904000	Gate3[9]	$924000
Gate3[2]	$908000	Gate3[10]	$928000
Gate3[3]	$90C000	Gate3[11]	$92C000
Gate3[4]	$910000	Gate3[12]	$930000
Gate3[5]	$914000	Gate3[13]	$934000
Gate3[6]	$918000	Gate3[14]	$938000
Gate3[7]	$91C000	Gate3[15]	$93C000

Note

The base address is typically stated in the hardware reference manual of the MACRO hardware device (i.e. ACC-5E3). It can be found by subtracting **Gate3[i].a** from **Sys.piom**.

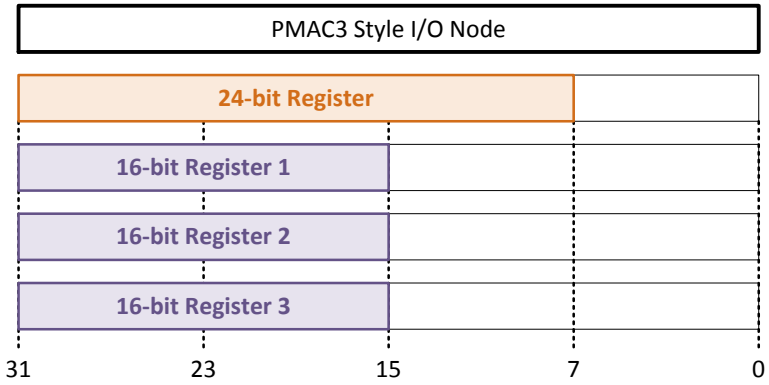
And the data registers' offsets for each I/O node:

		Data Register							
	I/O Node	24-bit		1st 16-bit		2nd 16-bit		3rd 16-bit	
		In	Out	In	Out	In	Out	In	Out
Bank A	2	$420	$520	$424	$524	$428	$528	$42C	$52C
	3	$430	$530	$434	$534	$438	$538	$43C	$53C
	6	$460	$560	$464	$564	$468	$568	$46C	$56C
	7	$470	$570	$474	$574	$478	$578	$47C	$57C
	10	$4A0	$5A0	$4A4	$5A4	$4A8	$5A8	$4AC	$5AC
	11	$4B0	$5B0	$4B4	$B4	$B8	$5B8	$4BC	$5BC
Bank B	2	$620	$720	$624	$724	$628	$728	$62C	$72C
	3	$630	$730	$634	$734	$638	$738	$63C	$73C
	6	$660	$760	$664	$764	$668	$768	$66C	$76C
	7	$670	$770	$674	$774	$678	$778	$67C	$77C
	10	$6A0	$7A0	$6A4	$7A4	$6A8	$7A8	$6AC	$7AC
	11	$6B0	$7B0	$6B4	$7B4	$6B8	$7B8	$6BC	$7BC

The data registers' offsets are found by subtracting **Gate3[i].MacroInA[j][k].a** (In or Out) from **Gate3[i].a.**

Using the ACC-65M with PMAC3 Address Offsets

With the PMAC3 Style MACRO IC, the MACRO I/O data is found in the upper fields:



Example: mapping the ACC-65M data, with address offsets, into PMAC3 Style MACRO IC 0, Bank **A,** node **2** requires adding the base address to the offset ($900000 + offset) of the desired data register.

Knowing that the general purpose inputs and outputs are in the 24-bit data register, the analog ADC inputs and DAC outputs in the 1st and 2nd 16-bit data registers, and the general purpose relays in the 3rd 16-bit data register (bits 27, and 28):

```
PTR Inputs->U.IO:$900420.8.24;        // GP Inputs
PTR Outputs->U.IO:$900520.8.24;       // GP Outputs

PTR ADC1->S.IO:$900424.16.16;         // Analog ADC 1 Input
PTR ADC2->S.IO:$900428.16.16;         // Analog ADC 2 Input

PTR DAC1->S.IO:$900524.16.16;         // Analog DAC 1 Output
PTR DAC2->S.IO:$900528.16.16;         // Analog DAC 2 Output

PTR GPRelay1->U.IO:$90052C.27.1;      // GP Relay Output 1
PTR GPRelay2->U.IO:$90052C.28.1;      // GP Relay Output 2
```

With the PMAC3 Style MACRO IC, all the data can be read and written to at will without further processing. No image word(s) shifting, or scaling required.

Note

Explicit address offsets mapping is useful for older Power PMAC firmware versions.

Note

Power PMAC users with firmware versions **1.5.8.305** or newer are highly encouraged to use the structure element addressing aforementioned in this manual.

PMAC2 Style ASIC

The Power PMAC can interface with up to 32 PMAC2 Style ASICs which base addresses are:

Index	Card I/O Address	Index	Card I/O Address
Gate2[0]	$800000	Gate2[16]	$840000
Gate2[1]	$804000	Gate2[17]	$844000
Gate2[2]	$808000	Gate2[18]	$848000
Gate2[3]	$80C000	Gate2[19]	$84C000
Gate2[4]	$810000	Gate2[20]	$850000
Gate2[5]	$814000	Gate2[21]	$854000
Gate2[6]	$818000	Gate2[22]	$858000
Gate2[7]	$81C000	Gate2[23]	$85C000
Gate2[8]	$820000	Gate2[24]	$860000
Gate2[9]	$824000	Gate2[25]	$864000
Gate2[10]	$828000	Gate2[26]	$868000
Gate2[11]	$82C000	Gate2[27]	$86C000
Gate2[12]	$830000	Gate2[28]	$870000
Gate2[13]	$834000	Gate2[29]	$874000
Gate2[14]	$838000	Gate2[30]	$878000
Gate2[15]	$83C000	Gate2[31]	$87C000

Note

The base address is typically stated in the hardware reference manual of the MACRO hardware device (i.e. ACC-5E). It can be found by subtracting **Gate2[i].a** from **Sys.piom**.

And the data registers' offsets for each I/O node:

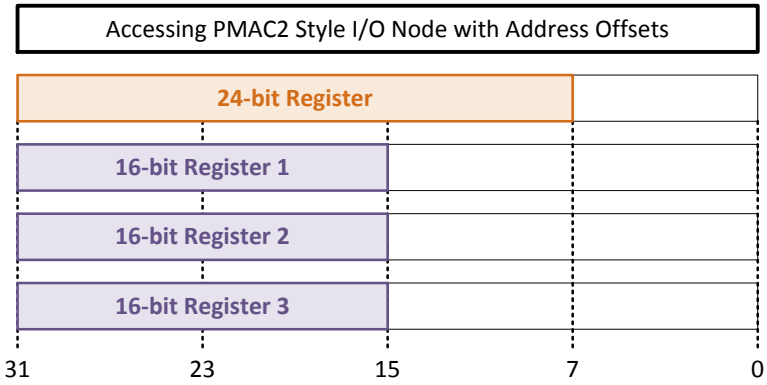
	2	3	6	7	10	11
24-bit	$120	$130	$160	$170	$1A0	$1B0
1st 16-bit	$124	$134	$164	$174	$1A4	$1B4
2nd 16-bit	$128	$138	$168	$178	$1A8	$1B8
3rd 16-bit	$12C	$13C	$16C	$17C	$1AC	$1BC

Note

The data registers' offsets are found by subtracting **Gate2[i].Macro[j][k].a** from **Gate2[i].a**

Using the ACC-65M with PMAC2 Address Offsets

When using explicit address offsets, the MACRO I/O data is found in the upper fields:



Example: mapping the ACC-65M data, with address offsets, into PMAC2 Style MACRO IC 0, node **2** requires adding the base address to the offset ($800000 + offset) of the desired data register.

Knowing that the general purpose inputs and outputs are in the 24-bit data register, the analog ADC inputs and DAC outputs in the 1st and 2nd 16-bit data registers, and the general purpose relays in the 3rd 16-bit data register (bits 27, and 28):

```
PTR Inputs->U.IO:$800120.8.24;        // GP Inputs
PTR Outputs->U.IO:$800120.8.24;       // GP Outputs

PTR ADC1->S.IO:$800124.16.16;         // Analog ADC 1 Input
PTR ADC2->S.IO:$800128.16.16;         // Analog ADC 2 Input

PTR DAC1->S.IO:$800124.16.16;         // Analog DAC 1 Output
PTR DAC2->S.IO:$800128.16.16;         // Analog DAC 2 Output

PTR GPRelay1->U.IO:$80012C.27.1;      // GP Relay Output 1
PTR GPRelay2->U.IO:$80012C.28.1;      // GP Relay Output 2
```

- The general purpose inputs can be bitwise mapping directly and read at will.
- The general purpose outputs can be mapped directly and written to. However, they do require an image word to allow writing to multiple bits simultaneously and reporting the state of each output.
- The analog ADC inputs can be read at will, they do not require further processing.
- The analog DAC outputs require an image word to report the written values.
- The GP relay outputs can be read and written to separately. An image word is required to allow writing to both outputs simultaneously.

Note

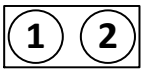
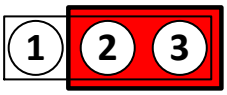
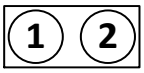
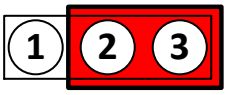
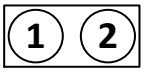
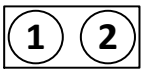
Explicit address offsets mapping is useful for older Power PMAC firmware versions.

Note

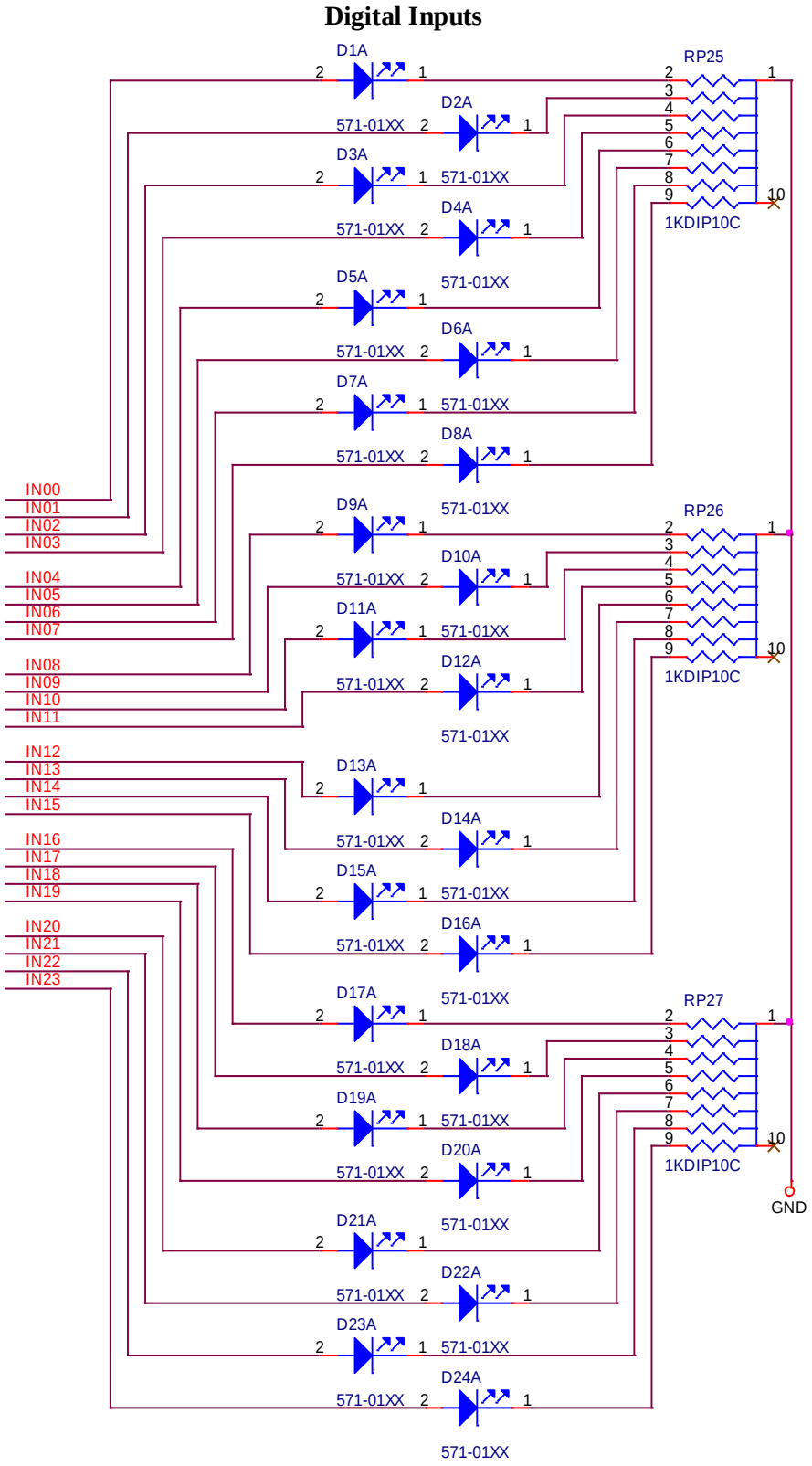
Power PMAC users with firmware versions **1.5.8.305** or newer are highly encouraged to use the structure element addressing aforementioned in this manual.

APPENDIX B: E-POINT JUMPERS

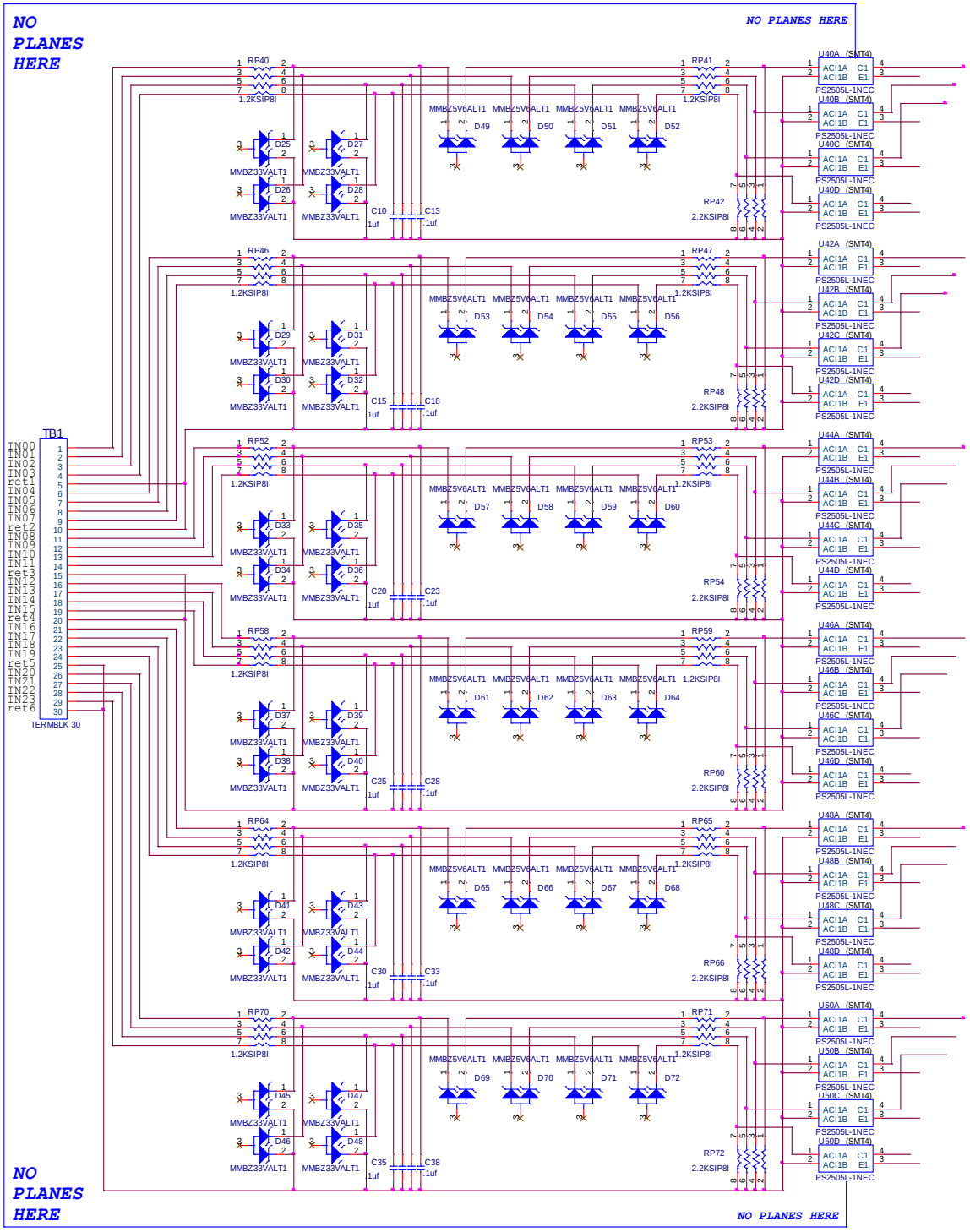
The ACC-65M jumpers are for internal use and set by the factory.

Jumper	Configuration	Default
E1: 	➢ Remove jumper to enable the watchdog timer ➢ Install jumper to disable watchdog timer (not advised)	NOT Installed
E2: 	➢ Jump pins 1 and 2 for firmware download through USB port. ➢ Jump pins 2 and 3 for normal operation.	2 – 3
E3: 	➢ Jump pins 1 and 2 for 9600-baud serial port operation. ➢ Remove jumper for 38400-baud serial port operation.	NOT Installed
E4: 	➢ Jump pins 1 and 2 for RJ-45 connection. ➢ Jump pins 2 and 3 for fiber optic connection.	Factory Set
JP1: 	➢ Remove jumper for ACC-65M. ➢ Install jumper for ACC-68M.	NOT Installed
JP2 – JP7	Reserved for Future Use.	N/A
JP7: 	➢ Jump pins 1 and 2 for re-initialization on power-up/reset. ➢ Remove jumper for normal operation	NOT Installed

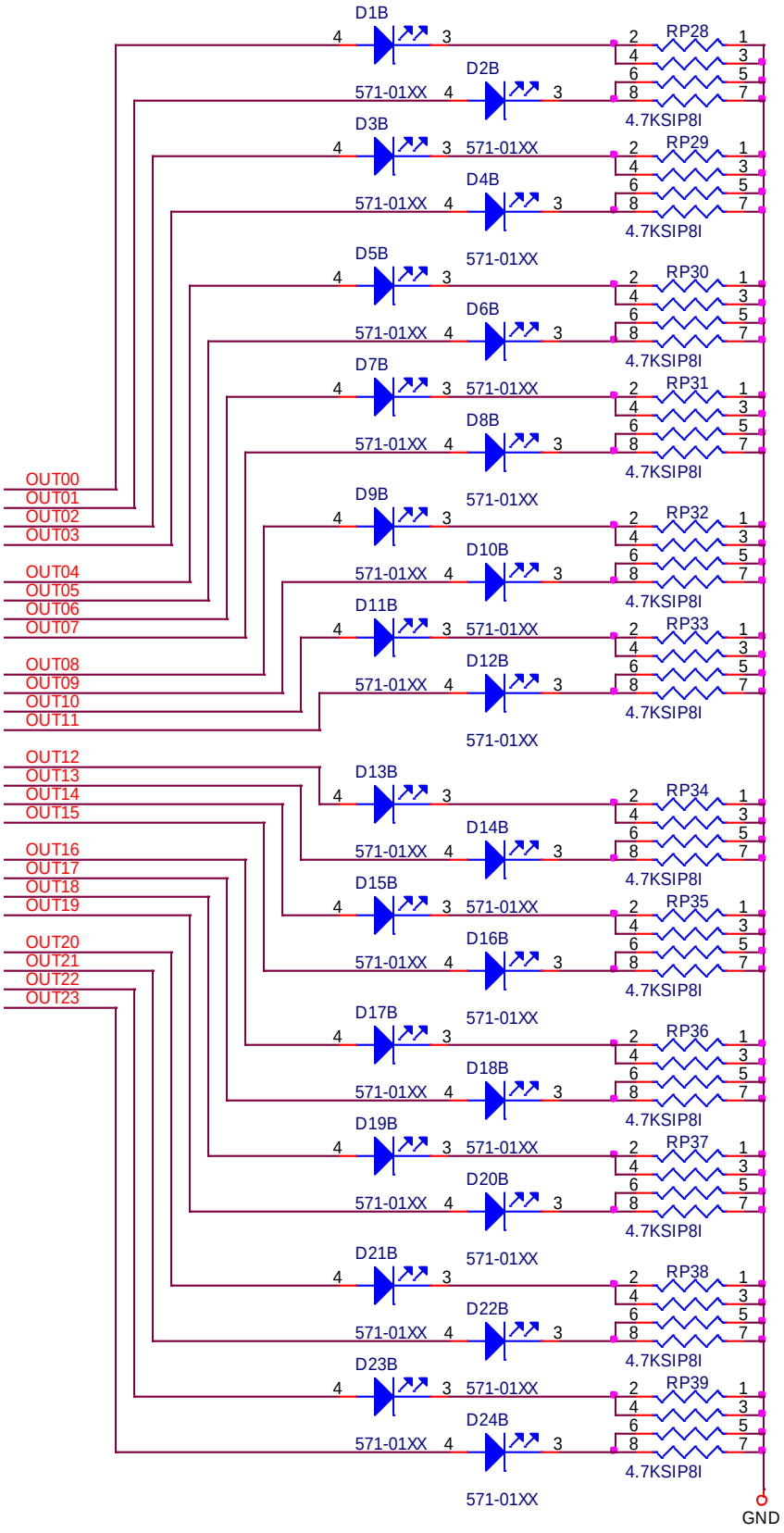
APPENDIX C: SCHEMATICS



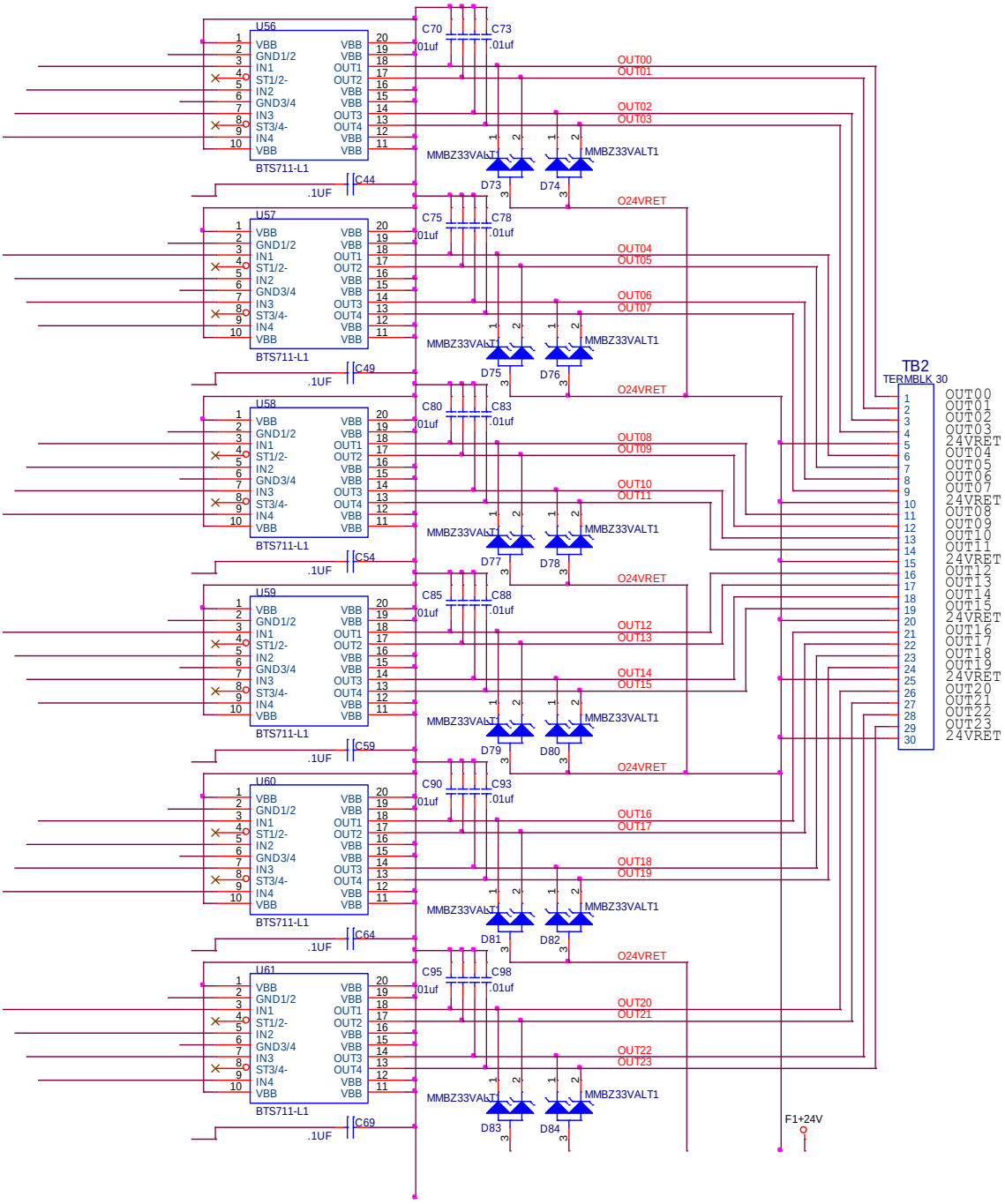
Digital Inputs (continued)



Digital Outputs



Digital Outputs (continued)



ADC/DAC/Relays Connector (OPT-1 Only)